(19)日本国特許庁（ＪＰ）　(12)公開特許公報（Ａ）　(11)特許出願公開番号
特開2000－228768
（P2000－228768A）
(43)公開日　平成12年８月15日(2000.8.15)

(51)Int.Cl.⁷	識別記号	ＦＩ	テーマコード（参考）
Ｈ０４Ｎ　7/24		Ｈ０４Ｎ　7/13	Ｚ　５Ｃ０５９

審査請求　未請求　請求項の数５　ＯＬ　（全 27 頁）

(21)出願番号　特願平11－29377

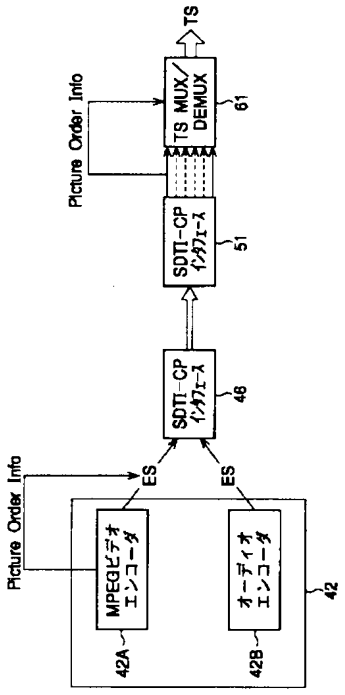
(22)出願日　平成11年２月５日(1999.2.5)

(71)出願人　000002185
ソニー株式会社
東京都品川区北品川６丁目７番35号
(72)発明者　田原　勝己
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(72)発明者　村上　芳弘
東京都品川区北品川６丁目７番35号　ソニー株式会社内
(74)代理人　100082131
弁理士　稲本　義雄

最終頁に続く

(54)【発明の名称】　デジタル信号伝送装置および方法、並びに提供媒体

(57)【要約】
【課題】　所定のスタジオと他のスタジオの両方において、画像データを迅速に処理できるようにする。
【解決手段】　MPEGビデオエンコーダ４２Ａは、エンコードしたビデオデータのエレメンタリストリームに、画像の符号化順序と表示順序を表すPicture OrderInformationを重畳し、SDTI-CPインタフェース４６を介して、同一スタジオ内のSDTI-CPネットワークに伝送する。TS MUX/DEMUX６１は、SDTI-CPインタフェース４６，５１を介して、エレメンタリストリームの供給を受けたとき、そこに含まれているPicture Order Informationを抽出し、それに基づいてタイムスタンプを生成し、トランスポートストリームに重畳して、他のスタジオに伝送する。



1

【特許請求の範囲】

【請求項１】　画像信号の符号化順序と、前記画像信号の表示順序に関する順序情報が、前記画像信号のアクセスユニット単位で挿入されている第１のビットストリームを受信する第１の受信手段と、

前記第１の受信手段により受信された前記第１のビットストリームから前記順序情報を抽出し、抽出した前記順序情報に基づいてタイムスタンプを生成し、第２のビットストリームに挿入して出力する第１の出力手段とを含むことを特徴とするデジタル信号伝送装置。

【請求項２】　前記第２のビットストリームを受信する第２の受信手段と、

前記第２の受信手段により受信された前記第２のビットストリームから、前記タイムスタンプを生成し、前記第１のビットストリームに挿入して出力する第２の出力手段とをさらに含むことを特徴とする請求項１に記載のデジタル信号伝送装置。

【請求項３】　前記符号化順序は、前記画像信号のフィールドの表示時間を単位とすることを特徴とする請求項１に記載のデジタル信号伝送装置。

【請求項４】　画像信号の符号化順序と、前記画像信号の表示順序に関する順序情報が、前記画像信号のアクセスユニット単位で挿入されている第１のビットストリームを受信する受信ステップと、

前記受信ステップの処理で受信された前記第１のビットストリームから前記順序情報を抽出し、抽出した前記順序情報に基づいてタイムスタンプを生成し、第２のビットストリームに挿入して出力する出力ステップとを含むことを特徴とするデジタル信号伝送方法。

【請求項５】　画像信号の符号化順序と、前記画像信号の表示順序に関する順序情報が、前記画像信号のアクセスユニット単位で挿入されている第１のビットストリームを受信する受信ステップと、

前記受信ステップの処理で受信された前記第１のビットストリームから前記順序情報を抽出し、抽出した前記順序情報に基づいてタイムスタンプを生成し、第２のビットストリームに挿入して出力する出力ステップとを含む処理をデジタル信号伝送装置に実行させるコンピュータが読み取り可能なプログラムを提供することを特徴とする提供媒体。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】本発明は、デジタル信号伝送装置および方法、並びに提供媒体に関し、特に、異なるスタジオ間で画像を送受信し、処理する場合において、送受信する画像信号のシステム全体における遅延量をできるだけ少なくするようにした、デジタル信号伝送装置および方法、並びに提供媒体に関する。

【０００２】

【従来の技術】画像信号または音声信号を符号化して伝

2

送する場合、ISO/IEC11172(MPEG-１)もしくはISO/IEC13818(MPEG-2)に示されている符号化方式が用いられることが多い。画像信号を、MPEG（Moving Picture Experts Group）方式で符号化する場合の技術として、コーディングフェーズ（coding phase）と両方向予測があげられる。

【０００３】coding phaseとは、図１に示すように、１枚の画像の画素エリア１のうち、符号化が行われる範囲としての有効画素エリア２を規定するためのコードであり、具体的には、全てのラインのうち、有効画素エリア２の最初のライン（図１のV-phase [line] の矢印で示されたライン）、および、全てのサンプル（画素）のうち、有効画素エリア２の最初のラインの最初のサンプル（図１のH-phase[sample]の矢印で示されたサンプル（画素））を示すものである。画素エリア１の最後のラインを、V-Phase[Lmax]とし、画素エリア１のラインの最後のサンプル（画素）を、H-Phase[Smax]とするとき、符号化は、例えば、垂直方向に、V-Phase[line]から、（V-Phase[Lmax] - V-Phase[line]+1）までの範囲で行われ、水平方向に、H-Phase[sample]から、（H-Phase[Smax]-H - Phase[sample]+1）までの範囲で行われる。

【０００４】また、一般の例えば、NTSC方式のテレビジョン受像機などの画像信号において、図１の画像信号のフレームは、図２に示すように、２枚のフィールド（図２のフィールド１およびフィールド２）から構成されている。フィールド１には、例えば、奇数ラインのデータが表示され、フィールド２には偶数ラインのデータが表示される。２枚のフィールドは、それぞれ、補助データ（ancillary data）と、画像（イメージ）データ（image data）から構成されている。

【０００５】補助データは、耳の不自由な人のための文字放送用のテレテキストデータ、タイムコード、または映画などの字幕のクローズドキャプションデータに利用され、ブランキング区間（例えば、各フィールドの第１０ライン乃至第２２ライン）（図２においては、いずれのフィールドも、そのフィールド内の上から順番の番号でラインが表されている）のうちの所定のラインに挿入されている。一般に、各フィールドの第２３ラインより図中下に位置するラインの部分（実際に、画像として表示される部分）は、画像データとして、MPEG方式などにより符号化される。

【０００６】前述したMPEG符号化方式は、画像データにのみ適用され、coding phase、および補助データについては、MPEGの規格には、特に明確に記述（規定）されていない。そのため、coding phaseは、自由度を有し、様々なアプリケーションによって異なっている。

【０００７】このような画像信号を、符号化または復号するときのシステム構成を、図３を参照して説明する。アプリケーションＡの画像データは、アプリケーション

Ａ用MPEGエンコーダ１１で符号化され、エレメンタリストリーム（ES）として出力され、アプリケーションＡ用MPEGデコーダ１２で復号される。アプリケーションＢの画像データは、アプリケーションＢ用MPEGエンコーダ１３で符号化され、エレメンタリストリーム（ES）として出力され、アプリケーションＢ用MPEGデコーダ１４で復号される。

【０００８】つまり、あるアプリケーションの画像データは、そのアプリケーション専用のエンコーダ（例えば、アプリケーションＡ用MPEGエンコーダ１１）で符号化され、エレメンタリストリーム（ES）として、そのアプリケーションに対応した専用のデコーダ（例えば、アプリケーションＡ用MPEGデコーダ１２）に出力される。専用デコーダに入力されたエレメンタリストリーム（ES）は、そのデコーダが有している、そのアプリケーションのcoding phaseに関する情報に基づき、復号される。

【０００９】次に、現在主流となっている補助データの伝送方式について、図４を参照して説明する。画像信号は、MPEGエンコーダ２１に入力され、分離部２１において、画像データと補助データに分離される。MPEGエンコード部２３は、画像データをMPEG符号化する。補助データは、可変長符号化部２４で、MPEGエンコード部２３より出力されたMPEG方式のトランスポートストリーム中のuser dataに挿入される。このトランスポートストリーム中のuser dataは、符号化された画像データ（ピクチャ）単位に挿入（記述）できるため、補助データは、対応する各符号化画像データのフレームのuser dataごとに挿入される。

【００１０】補助データが挿入されたuser dataを含むトランスポートストリームは、所定の伝送路を介して伝送され、MPEGデコーダ２５に入力される。MPEGデコーダ２５内の可変長復号部２６は、補助データと画像データとを分離する。画像データはMPEGデコード部２７で復号され、合成部２８で補助データと合成されて、画像信号として、図示せぬ表示装置に出力される。

【００１１】次に、MPEG方式の予測について説明する。両方向予測に基づいて生成された符号化画像データのピクチャはＢピクチャと称される。Ｂピクチャは、時間的に前または後に位置する２枚の参照画像データから予測されて生成される。前方予測に基づいて生成された符号化画像データのピクチャはＰピクチャと称される。Ｐピクチャは、時間的に前に位置する１枚の参照画像データから予測されて生成される。予測が行われず、そのまま画像データが符号化（イントラ符号化）された符号化画像データのピクチャはＩピクチャと称される。つまり、入力された画像データは、Ｂピクチャ、Ｐピクチャ、またはＩピクチャのいずれかの符号化画像データに符号化される。

【００１２】両方向予測および前方予測について、図５を参照して説明する。図５の例では、１つのGOP（Group of Picture）が、９枚のピクチャから構成されている。図５の上段は、符号化画像データが生成されるときの、予測の方向（依存関係）を示す予測構造を表しており、図５の下段は、実際に画像データが符号化される順序を示す符号化構造を表している。Ｂピクチャ、またはＰピクチャは、時間的に前または後の画像データから予測されて生成されるために、Ｂピクチャ、またはＰピクチャだけでは、符号化を行うことはできない。すなわち、Ｂピクチャ、またはＰピクチャは、参照画像データとの差分をデータとする符号化画像データであるので、Ｂピクチャ、またはＰピクチャだけでは、画像データを復号することもできない。

【００１３】予測の依存関係について詳細に説明すると、例えば、GOP(N-1)において、表示順序が先頭のＢピクチャは、先頭から３番目のＩピクチャと、図示されていないGOP(N-2)（GOP(N-1)の直前のGOP）の最後のＰピクチャから予測されて符号化される。先頭から２番目のＢピクチャも同様に、先頭から３番目のＩピクチャと、図示されていないGOP(N-2)の最後のＰピクチャから予測されて符号化される。先頭から３番目のＩピクチャは、そのまま符号化（イントラ符号化）される。先頭から４番目、および５番目のＢピクチャは、先頭から３番目のＩピクチャと先頭から６番目のＰピクチャから予測されて符号化される。先頭から６番目のＰピクチャは、先頭から３番目のＩピクチャから予測されて符号化される。

【００１４】つまり、予測構造において、Ｂピクチャ（例えば、先頭のＢピクチャと先頭から２番目のＢピクチャ）が符号化されるには、予測の参照画像データ（例えば、３番目のＩピクチャとGOP(N-2)の最後のＰピクチャ）が、先に符号化されている必要がある。すなわち、図５の下段に示すような、符号化構造の順番で符号化されなければならない。そのため、MPEGエンコーダは、符号化（エンコード）時、ＩピクチャとＰピクチャの間に存在する、２枚の連続したＢピクチャを符号化するために必要な参照画像データ（先頭から３番目のＩピクチャ）が入力されるまで、２枚のＢピクチャをバッファリングする（Ｉピクチャが入力されるまで、符号化の開始を遅延させる）必要がある。このバッファリングにより、MPEGエンコーダにおいて、入力された画像信号を符号化するとき、参照画像データに挟まれたＢピクチャの枚数（＝２）＋１枚（合計３枚）の遅延時間が発生する。

【００１５】このMPEGエンコーダにおいて発生する遅延について、図６を参照して、さらに詳細に説明する。図６の上段はMPEGエンコーダ３１における、入力画像（画像データ）の入力順番（表示順序）とその種別を表しており、図６の中段は、入力画像（画像データ）が符号化された符号化画像データの順番を表している。

【００１６】MPEGエンコーダ３１は、時刻ｔ４のＰピク

チャが入力されるまで（Ｂピクチャの枚数（＝２）＋１枚分の時間だけ）、時刻ｔ１に入力されたＩピクチャの符号化を遅延させる（前述のバッファリングを行う）。すなわち、MPEGエンコーダ３１は、時刻ｔ４で時刻ｔ１に入力されたＩピクチャを符号化し、時刻ｔ５で時刻ｔ４に入力されたＰピクチャを符号化し、時刻ｔ６で時刻ｔ２に入力されたＢピクチャを符号化し、時刻ｔ７で時刻ｔ３に入力されたＢピクチャを符号化する。その後、MPEGエンコーダ３１は、入力画像を図６の中段の符号化画像データの順番で、順次、符号化する。

【００１７】このように、MPEGエンコーダ３１は、入力画像に対して、予測に必要な参照画像データから順次符号化する。つまり、MPEGエンコーダ３１において、入力画像（画像データ）は、入力画像の順番から、符号化される画像の順番に並べ変えられ、図６の中段に示すように、入力画像が符号化された順番にビットストリームとして、MPEGデコーダ３２に出力される。

【００１８】このように、入力画像が符号化される順番と表示される順番は一致しないので、MPEGにおいては、符号化順序を表すDTS（Decoding Time Stamp）と、表示順序を表すPTS（Presentation Time Stamp）が、トランスポートストリーム中に挿入されるようになされている。入力画像の符号化順序と表示順序の関係をさらに説明すると、いま、符号化順序をフレーム単位で表すものとすると、図６に示すように、入力画像は符号化される順番で付番され、時刻ｔ４で符号化されたＩピクチャの符号化順序の値は”１”となり、時刻ｔ５で符号化されたＰピクチャの符号化順序の値は”２”となり、時刻ｔ６で符号化されたＢピクチャの符号化順序の値は”３”となる。以下、同様に符号化される順番に符号化順序が付番される。DTSはフレーム単位で符号化順序を表しているわけではないが、ほぼ、この符号化順序に対応する。DTSはまた、ビットストリームを音声信号などと多重化して出力するとき、MPEGデコーダ３２において、復号する順番として使用される。

【００１９】表示順序は、画像データが復号されて表示される順番（入力画像データの順番と同じ）となる。具体的には、時刻ｔ４で符号化されるＩピクチャ（符号化順序＝１）の表示順序は、時刻ｔ５でＰピクチャ（符号化順序＝２）が符号化されるときに表示されなければならないので、そのＰピクチャの符号化順序と同じ値”２”となる。時刻ｔ５で符号化されるＰピクチャ（符号化順序＝２）の表示順序は、時刻ｔ８でＰピクチャ（符号化順序＝５）が符号化されるとき表示されなければならないので、そのＰピクチャの符号化順序と同じ値”５”となる。時刻ｔ６で符号化されるＢピクチャ（符号化順序＝３）は、符号化されると、直ちに復号されて表示されなければならないので、符号化順序と表示順序は同一となり、表示順序の値は”３”となる。時刻ｔ７で符号化されるＢピクチャ（符号化順序＝４）も同様に、表示順序＝符号化順序＝４となる。PTSはフレーム単位で表示順序を表しているわけではないが、ほぼ、この表示順序に対応する。PTSはまた、MPEGデコーダ３２において、復号後、出力（表示）する順番として使用される。

【００２０】MPEGデコーダ３２では、符号化順序が連続する２枚のＩピクチャもしくはＰピクチャのうち、最初の１枚目は、２枚目が復号されるとき、表示される。例えば、MPEGエンコーダ３１が出力したビットストリーム中の符号化画像データの１枚目のＩピクチャ（符号化順序＝１，表示順序＝２）と、２枚目のＰピクチャ（符号化順序＝２，表示順序＝５）は、符号化順序が連続しており、２枚目のＰピクチャが符号化されるとき（時刻ｔ５のとき）、１枚目のＩピクチャが復号されて表示される。

【００２１】このように、MPEGエンコーダ３１は、入力画像（画像データ）の予測構造（参照画像データに挟まれているＢピクチャの枚数）を知っているので、画像データに符号化順序（DTS）と表示順序（PTS）を付番することができる。

【００２２】放送局などのスタジオ内で、図７に示すシステム構成で、MPEG符号化されたビットストリームを伝送することが考えられている。スタジオ４１内のMPEGエンコーダ４２、MPEGエンコーダ４３、MPEGデコーダ４４、およびMPEGデコーダ４５は、それぞれ、SDTI-CP（Serial Data Transfer Interface - Content Package）インタフェース４６乃至４９を介して、SDTI-CPネットワーク（例えば、同軸ケーブルにより構成されるネットワーク）５０に接続される。SDTI-CPネットワーク５０は、SDI（Serial Data Interface）をベースとした２７０Mbpsの伝送速度を有し、MPEG方式のエレメンタリストリーム（ES）をそのまま伝送することが可能であり、スタジオ内のような閉じたネットワークに適している。

【００２３】スタジオ４１において、例えば、MPEGエンコーダ４２は、MPEG符号化されたエレメンタリストリーム（ES）を、SDTI-CPネットワーク５０を介して、MPEGデコーダ４４、MPEGデコーダ４５に伝送することができる。

【００２４】このSDTI-CPネットワーク５０で伝送されるエレメンタリストリーム（ES）は、図８に示す構造となっており、画像信号のフレーム単位で、画像データ（図８において薄い影を付けた部分）と音声データ（図８において濃い影を付けた部分）がパッキングされており、フレームシンク（図８の点線）により区切られたフレーム境界で、簡単に編集を行うことができる。このエレメンタリストリーム（ES）における画像データと音声データは、イントラ（フレーム内符号化）処理されたデータである。

【００２５】図７に示すスタジオ４１のシステムは、そのMPEGエンコーダ４２，４３とMPEGデコーダ４４，４５

が、SDTI-CPインタフェース４６乃至４９を介して、SDTI-CPネットワーク５０に接続されており、図３に示したような、アプリケーションごとに専用のエンコーダ（例えば、アプリケーションＡ用MPEGエンコーダ１１）とデコーダ（例えば、アプリケーションＡ用MPEGデコーダ１２）が１対１に対応するシステムとは構成が異なっている。すなわち、デコーダ（例えば、図７のMPEGデコーダ４４）は、エンコーダ（例えば、図７のMPEGエンコーダ４２またはMPEGエンコーダ４３）で、様々なアプリケーションの画像信号が符号化された、エレメンタリストリーム（ES）を受け取ることができる。

【００２６】図７のシステム構成を、図３のシステム構成に対応させた図９を参照して説明すると、アプリケーションＡ用MPEGエンコーダ４２で符号化されたエレメンタリストリーム（ES）と、アプリケーションＢ用MPEGエンコーダ４３で符号化されたエレメンタリストリーム（ES）は、それぞれSDTI-CPインタフェース４６，４７に入力される。SDTI-CPインタフェース４６，４７は、それぞれMPEG符号化されたエレメンタリストリーム（ES）を、SDTIフォーマットのエレメンタリストリーム（ES）に変換して、SDTI-CPネットワーク５０を介して伝送する。SDTI-CPインタフェース４８は、SDTIフォーマットのエレメンタリストリーム（ES）を、MPEG符号化されたエレメンタリストリーム（ES）に変換し、MPEGデコーダ４４に出力する。

【００２７】MPEGデコーダ４４は、それぞれ入力されたアプリケーションＡ用のエレメンタリストリーム（ES）とアプリケーションＢ用のエレメンタリストリーム（ES）を復号する。

【００２８】ところで、補助データがトランスポートストリーム中のuser dataに挿入される場合、補助データは、対応する符号化画像のフレーム単位で挿入されるため、その挿入は１フレーム（２フィールド）ごととなる。

【００２９】符号化される信号が、３－２プルダウン処理（例えば、２４Hzのフレームレートを持つ映画の画像信号を、３０Hzのフレームレートを持つNTSC方式の画像信号に変換する処理）された信号である場合、その信号は、図１０に示すように、２４Hzの各フレームを、交互に、リピートフィールドが作成されていない２フィールドのフレーム、またはリピートフィールドが作成されている３フィールドのフレームとすることで、３０Hzの信号とされている。

【００３０】例えば、図４のMPEGエンコーダ２１は、３－２プルダウン処理により３０Hzのフレームレートに変換された画像信号が入力されたとき、フィールドの繰り返しを検出して、元の２４Hzの符号化フレーム単位で符号化を行い、その処理に対応して、フラグ（Repeat_first_field，Top_field_first）を生成する。

【００３１】Repeat_first_fieldのフラグの”１”は、リピートフィールドが作成されたことを意味し、Repeat_first_fieldのフラグの”０”は、リピートフィールドが作成されていないことを意味する。Top_field_firstのフラグは、フレームを構成するフィールドのうち、最初のフィールドがトップフィールドであるのか、またはボトムフィールドであるのかを表している。Top_field_firstのフラグの”１”は、トップフィールドがボトムフィールドより時間的に早いフレーム構造であることを表しており、Top_field_firstのフラグの”０”は、ボトムフィールドがトップフィールドより時間的に早いフレーム構造であることを表している。

【００３２】図１０に示すように、原信号のフレームが３フィールドのフレームである場合、対応する１つの符号化フレームには、同一位相の２枚のフィールド（３－２プルダウン処理によりコピーにより生成されたフィールドとそのコピー元のフィールド）が存在する。しかしながら、この元の１つのフレームを構成する３フィールドが符号化フレーム単位で符号化されるとき、各符号化フレームごとの補助データは、符号化フレーム単位で、１つの補助データとして、user dataに記述されるため、原信号の同一位相のフィールドに異なった補助データが記述されていても、その異なった補助データを区別することができなくなってしまう。

【００３３】

【発明が解決しようとする課題】例えば、所定のスタジオで、MPEG方式でエンコードした画像を処理し、他のスタジオにトランスポートストリームとして伝送し、他のスタジオでも処理するような場合、画像データの伝送は、符号化された順序で行う必要があるが、トランスポートストリームとして画像データを伝送する装置は、通常、符号化順序を知らないので、入力された画像データを一旦バッファリングし、符号化順序を知る必要がある。その結果、バッファリングのために画像データの供給を受けてから、それを伝送するまでに時間がかかり、迅速な処理、特に、リアルタイム処理が困難となる課題があった。

【００３４】本発明はこのような状況に鑑みてなされたものであり、画像データを迅速に処理することができるようにするものである。

【００３５】

【課題を解決するための手段】請求項１に記載のデジタル信号伝送装置は、画像信号の符号化順序と、画像信号の表示順序に関する順序情報が、画像信号のアクセスユニット単位で挿入されている第１のビットストリームを受信する第１の受信手段と、第１の受信手段により受信された第１のビットストリームから順序情報を抽出し、抽出した順序情報に基づいてタイムスタンプを生成し、第２のビットストリームに挿入して出力する第１の出力手段とを含むことを特徴とする。

【００３６】請求項４に記載のデジタル信号伝送方法

は、画像信号の符号化順序と、画像信号の表示順序に関する順序情報が、画像信号のアクセスユニット単位で挿入されている第１のビットストリームを受信する受信ステップと、受信ステップの処理で受信された第１のビットストリームから順序情報を抽出し、抽出した順序情報に基づいてタイムスタンプを生成し、第２のビットストリームに挿入して出力する出力ステップとを含むことを特徴とする。

【００３７】請求項５に記載の提供媒体は、画像信号の符号化順序と、画像信号の表示順序に関する順序情報が、画像信号のアクセスユニット単位で挿入されている第１のビットストリームを受信する受信ステップと、受信ステップの処理で受信された第１のビットストリームから順序情報を抽出し、抽出した順序情報に基づいてタイムスタンプを生成し、第２のビットストリームに挿入して出力する出力ステップとを含む処理をデジタル信号伝送装置に実行させるコンピュータが読み取り可能なプログラムを提供することを特徴とする。

【００３８】請求項１に記載のデジタル信号伝送装置、請求項４に記載のデジタル信号伝送方法、および請求項５に記載の提供媒体においては、受信された第１のビットストリームから順序情報が抽出され、抽出した順序情報に基づいてタイムスタンプが生成され、第２のビットストリームに挿入される。

【００３９】

【発明の実施の形態】図１１は、相互に離れた場所に位置するスタジオ４１とスタジオ７１の間で、MPEG符号化されたビットストリームを伝送するシステムの例を表しており、図７における場合と対応する部分には、同一の符号を付してある。この例の場合、各スタジオ内のネットワークを、多重化装置（以下、TS MUX/DEMUXと称する）を介して、衛星やATM（Asynchronous Transfer Mode）などの公衆網と接続することで、ビットストリームの伝送を行うことができる。スタジオ７１のMPEGエンコーダ７２乃至SDTI-CPインタフェース７９は、図７のスタジオ４１のMPEGエンコーダ４２乃至SDTI-CPインタフェース４９に対応するものであるので、ここでは、その説明を省略する。

【００４０】SDTI-CPネットワーク５０におけるエレメンタリストリーム（ES）は、図１２に示すように、TS MUX/DEMUX６１により、１８８バイト単位のトランスポートストリーム（TS）に変換されて所定の伝送媒体を介して伝送され、伝送されたトランスポートストリーム（TS）は、TS MUX/DEMUX６２で、SDTIフォーマットのエレメンタリストリーム（ES）に変換される。なお、図１２において、薄い影を付けた部分は画像データのパケットを示しており、濃い影を付けた部分は音声データのパケットを示しており、影をつけていない部分は空きデータのパケットを示している。

【００４１】図１１において、スタジオ４１内のMPEGエンコーダ４２の出力するエレメンタリストリーム（ES）が、TS MUX/DEMUX６１でトランスポートストリーム（TS）に変換されるまでを、図１３を参照して説明する。MPEGエンコーダ４２内のMPEGビデオエンコーダ４２Ａは、MPEG符号化された画像データのエレメンタリストリーム（ES）を、SDTI-CPインタフェース４６に出力し、オーディオエンコーダ４２Ｂは、音声データのエレメンタリストリーム（ES）を、SDTI-CPインタフェース４６に出力する。SDTI-CPインタフェース４６は、入力されたエレメンタリストリーム（ES）をSDTIベースのフォーマットのエレメンタリストリーム（ES）に変換し、SDTI-CPネットワーク５０を介して、SDTI-CPインタフェース５１に出力する。SDTI-CPインタフェース５１は、SDTIフォーマットのＥＳをMPEG符号化のエレメンタリストリーム（ES）に変換し、TS MUX/DEMUX６１に出力する。TS MUX/DEMUX６１は、MPEG符号化のエレメンタリストリーム（ES）を１８８バイト単位のトランスポートストリーム（TS）に変換して伝送媒体に出力する。

【００４２】図１１において、TS MUX/DEMUX６１から伝送されたトランスポートストリーム（TS）は、ATMなどの公衆網を介して、TS MUX/DEMUX６２に入力され、MPEG符号化のエレメンタリストリーム（ES）に変換される。SDTI-CPインタフェース８１において、MPEG符号化されたエレメンタリストリーム（ES）は、SDTIフォーマットのエレメンタリストリーム（ES）に変換され、スタジオ７１のSDTI-CPネットワーク８０に出力される。MPEGデコーダ７４は、SDTI-CPインタフェース７８を介して、スタジオ２１から伝送されたエレメンタリストリーム（ES）を受信することができる。

【００４３】図１１のシステムにおいて、スタジオ４１とスタジオ７１の間で、画像信号を伝送する場合、多重化装置（例えば、図１１のTS MUX/DEMUX６１）には、画像データが符号化されたエレメンタリストリーム（ES）のみが入力されるので、多重化装置は、エンコーダ（例えば、図１１のMPEGエンコーダ４２）において行われた符号化順序の並べ変えの情報を知らない。このため、多重化装置は、入力されたエレメンタリストリーム（ES）を解釈して、エレメンタリストリーム（ES）をトランスポートストリーム（TS）に変換する処理をしなければならない。

【００４４】このエレメンタリストリームを解釈する処理は、画像信号の符号化構造を把握する処理である。つまり、前述したエンコーダで入力画像をバッファリングして符号化し、ビットストリームに変換するときと同様の処理が、多重化装置でも必要となる。この多重化装置におけるバッファリング処理により、後段での画像データの復号までに遅延が発生し、リアルタイム処理が困難になる。

【００４５】この多重化装置で発生する、システム構成上、問題となる遅延について、図１４を用いて説明す

る。多重化装置に、図１４に示す符号化順序で、エンコーダで符号化されたビットストリームが入力された場合、多重化装置は、２枚の連続したＩピクチャ（時刻ｔ４）とＰピクチャ（時刻ｔ５）が入力された後、さらにそれらに挟まれた２枚のＢピクチャ（時刻ｔ６と時刻ｔ７）に続いてＰピクチャ（時刻ｔ８）が入力されるまで、表示順序を確定することができない。つまり、多重化装置は、図５に示したような符号化構造（ＩピクチャとＰピクチャの間に２枚のＢピクチャが挟まれている構造）を知らないので、２枚のＢピクチャが入力され、その次のＰピクチャ（時刻ｔ８）が入力されたとき（Ｂピクチャの入力が終了したことを確認したとき）、時刻ｔ４に入力されたＩピクチャの表示順序（PTS）を確定することができる。

【００４６】すなわち、この多重化装置においては、時刻ｔ８になって初めて、符号化順序＝１のＩピクチャの表示順序＝２を確定することができる。このため、多重化装置では、２枚の連続したＩピクチャまたはＰピクチャ＋２枚のＢピクチャの出現周期（＝４）の遅延が発生する。この遅延は、エンコーダでの遅延とは別に新たに発生するため、システム全体として、エンコーダの遅延（＝３）＋多重化装置の遅延（＝４）＝７の大幅な遅延が生じる。

【００４７】そこで本発明においては、符号化順序と表示順序を含む順序情報を、PictureOrder Infoとしてエレメンタリストリームに重畳するようにしている。この点のについては、後に詳述する。

【００４８】図１５は、本発明を適用したMPEGエンコーダ４２の構成例を表している。MPEGエンコーダ４２の分離部１０１は、入力された画像信号から、画像データと補助データを分離し、画像データをMPEGエンコード部１０３に出力し、補助データをエンコードコントローラ１０２に出力する。MPEGエンコード部１０３は、入力された画像データをMPEG方式により符号化するとともに、符号化順序を示すDTS_counter、および表示順序を示すPTS_counterを含むPOI（Picture Order Infomation）をエンコードコントローラ１０２に出力する。また、MPEGエンコード部１０３は、符号化した範囲の左上の位置（図１の有効画素エリア２の左上の画素の位置）を表すコーディングフェーズ（Coding Phase（V-Phase, H-Phase））をエンコードコントローラ１０２に供給する。

【００４９】エンコードコントローラ１０２は、分離部１０１より供給された補助データに、それが属するフィールドを識別するField IDと、それが挿入されているラインを表すLine_numberを付加し、その補助データと、MPEGエンコード部１０３より供給されたPOI、およびCoding Phase Information（CPI）を適宜処理し、user dataのフォーマットのデータとして、可変長符号化部１０４に出力し、多重化させる。

【００５０】可変長符号化部１０２は、MPEGエンコード部１０３より供給されたエンコードされた画像データを可変長符号化するとともに、エンコードコントローラ１０２より供給されたuser dataを画像データのエレメンタリストリームに挿入する。可変長符号化部１０４より出力されたエレメンタリストリームは、送信バッファ１０５を介して出力される。

【００５１】図１６は、本発明を適用したMPEGデコーダ４４の構成例を表している。受信バッファ１１１は、入力されたデータを一旦バッファリングした後、可変長復号部１１２に出力する。可変長復号部１１２は、入力されたデータから、画像データとuser dataとを分離し、画像データをMPEGデコード部１１４に出力し、user dataをデコードコントローラ１１３に出力する。デコードコントローラ１１３は、user dataからPOIとCPIを分離し、それらをMPEGデコード部１１４に出力する。また、デコードコントローラ１１３は、user dataから分離した補助データを合成部１１５に出力する。MPEGデコード部１１４は、可変長復号部１１２より入力された画像データを、デコードコントローラ１１３より入力されたPOIとCPIを参照してデコードし、デコードした結果を合成部１１５に出力している。合成部１１５は、デコードコントローラ１１３より供給された補助データをMPEGデコード部１１４より供給された画像データと合成し、出力する。

【００５２】以下に、MPEGエンコーダ４２とMPEGデコーダ４４の動作について説明する。

【００５３】CPIをuser dataに記述して、エレメンタリストリーム（ES）に重畳して出力する動作を、図１７を参照して説明すると、アプリケーションＡ用MPEGエンコーダ４２は、アプリケーションＡの画像信号を符号化したエレメンタリストリーム（ES）に、CPI（画像信号のフォーマットにおける有効画素エリアを示すデータであるV-PhaseとH-Phase）をuser dataに記述して（図１５の可変長符号化部１０４がエンコードコントローラ１０２から出力されるCPIのデータをuser dataに記述して）、SDTI-CPインタフェース４６に出力する。アプリケーションＢ用ＭＰＥＧエンコーダ４３も、ＭＰＥＧエンコーダ４２と同様に構成されており、アプリケーションＢの画像信号を符号化したエレメンタリストリーム（ES）に、CPIをuser dataとして記述して、SDTI-CPインタフェース４７に出力する。

【００５４】SDTI-CPインタフェース４６の構成例について、図１８を参照して説明する（他のSDTI-CPインタフェースも同様に構成されている）。復号部１２１は、MPEGエンコーダ４２から入力されたMPEG符号化されたエレメンタリストリーム（ES）を、符号化パラメータと画像データに分離し、画像データを復号し、符号化パラメータとともに符号化パラメータ多重化部１２２に出力する。符号化パラメータ多重化部１２２は、画像信号と符号化パラメータから、SDTI-CPベースのエレメンタリス

トリーム（ES）を生成して出力する。

【００５５】SDTI-CPインタフェース４６に、SDTI-CPベースのエレメンタリストリーム（ES）が入力された場合、符号化パラメータ分離部１２３は、エレメンタリストリーム（ES）から画像データと符号化パラメータを分離して、それぞれ符号化部１２４に出力する。符号化部１２４は、符号化パラメータを用いて、画像データを符号化し、MPEG符号化されたエレメンタリストリーム（ES）として出力するか、またはパケット化部１２５でMPEG符号化されたトランスポートストリーム（TS）として出力する。

【００５６】SDTI-CPインタフェース４６，４７は、MPEG符号化されたエレメンタリストリーム（ES）を、SDTIフォーマットのエレメンタリストリーム（ES）に変換して、SDTI-CPネットワーク５０を介して、伝送する。SDTI-CPインタフェース４８は、SDTIフォーマットのエレメンタリストリーム（ES）をMPEG符号化されたエレメンタリストリーム（ES）に変換し、MPEGデコーダ４４に出力する。

【００５７】MPEGデコーダ４４は、入力されたアプリケーションＡのエレメンタリストリーム（ES）またはアプリケーションＢのエレメンタリストリーム（ES）を復号し、それぞれのエレメンタリストリーム（ES）に記述されている（図１６のデコードコントローラ１１３が出力する）CPIに基づいて、画像信号を有効画素エリアに配置するように復号する。

【００５８】MPEGエンコーダ４２，４３は、それぞれが符号化したアプリケーションの画像信号のCPIをMPEG符号化されたエレメンタリストリーム（ES）中のuser dataに記述することにより、画像データとともに、CPIを伝送することができる。また、MPEGエンコーダ４２，４３は、CPIを伝送する機能を有することにより、様々なアプリケーションを符号化して伝送することができる。

【００５９】MPEGデコーダ４４は、MPEG符号化されたエレメンタリストリーム（ES）に挿入されているCPIを分離、解釈することにより、様々なCoding Phaseを有するアプリケーションの画像を、適切に、有効画素エリアに配置するように復号処理することができる。

【００６０】なお、本発明に実施の形態においては、CPIを、MPEG符号化されたエレメンタリストリームのuser dataに挿入したが、ビットストリームに他の方法で挿入するようにしてもよい。

【００６１】次に、MPEGエンコーダ４２において、各フィールドごとの複数の補助データを識別する動作について、図１９を参照して説明する。３－２プルダウン処理が実施されている符号化フレームの原信号（３０Hz）の各フィールドに、補助データが挿入されているものとする。この補助データを有する原信号（３０Hz）が、元のフレームレートの符号化フレーム（２４Hz）に符号化されるとき、各符号化フレームに含まれる２つまたは３つのフィールドに記述されている補助データに、符号化されたフィールドに対応した識別子が、field_ID（０乃至２のカウンタの値）として付加されて（図１５のエンコードコントローラ１０２で補助データにfield_IDが付加されて）、補助データともに伝送される。このfield_IDが補助データに付加されることにより、補助データがどの符号化フレーム内の、どのフィールドに対応したものであるかが識別される。

【００６２】具体的に説明すると、図１９の先頭の符号化フレームのフィールドの枚数は、２枚であるので、補助データは２つ存在する。符号化フレームは２枚のフィールドから生成され、その符号化フレームに対応する２枚のフィールドの、それぞれの補助データに、field_IDとして、”０”または”１”が付加される。

【００６３】先頭から２番目の符号化フレームのフィールドの枚数は、３枚であるので、補助データは３つ存在する。符号化フレームは３枚のフィールドから生成され、その符号化フレームに対応する３枚のフィールドの、それぞれの補助データに、field_IDとして、”０”，”１”または”２”が付加される。

【００６４】つまり、１枚の符号化フレームに対して、１つの補助データが生成されるのではなく、符号化フレームに含まれていたフィールドの枚数と同じ数の補助データが生成され、それぞれにfield_IDが付加される。その結果、同一の符号化フレームに含まれる複数の補助データは、付加されたfield_IDにより、符号化フレーム内で識別されるので、それぞれの補助データに異なった情報が含まれていても、識別できなくなることはない。

【００６５】MPEGエンコーダ４２では、画像データがMPEGエンコード部１０３でエンコードされ、可変長符号化部１０４で可変長符号化される。また、エンコードコントローラ１０２で補助データにfield_IDと、補助データが挿入されていたライン番号（Line_number）が付加され、エレメンタリストリーム中のuser data中に、ancillary dataとして挿入される。これにより、複数の補助データを識別して伝送することができる。

【００６６】MPEGデコーダ４４では、MPEG符号化されたエレメンタリストリーム中のuser dataが可変長復号部１１２で分離され、デコードコントローラ１１３に供給される。デコードコントローラ１１３は、user dataに挿入されているancillary dataのfield_IDとLine_numberに基づいて、複数の補助データを識別、分離し、合成部１１５に出力する。合成部１１５は、MPEGデコード部１１４で復号された画像データと、それに対応する補助データ（テキストデータ）を合成し、出力する。

【００６７】MPEGエンコーダ４２は、システム全体の遅延を少なくするために、図２０に示すような符号化順序と表示順序を管理するPOIを生成する。

【００６８】例えば、MPEGエンコーダ４２のMPEGエンコード部１０３に入力された画像データが、３－２プルダ

ウン処理により、２４Hzのフレームレートに変換された画像信号である場合、図２０（Ａ）に示すようなフラグ（Repeat_first_field, Top_field_first）により、各フレームが管理される。

【００６９】Repeat_first_fieldのフラグの”１”は、リピートフィールドを作成する必要があることを意味し、Repeat_first_fieldのフラグの”０”は、リピートフィールドを作成する必要がないことを意味する。Top_field_firstのフラグは、フレームを構成するフィールドのうち、最初のフィールドがトップフィールドであるのか、またはボトムフィールドであるのかを表している。Top_field_firstフラグの”１”は、トップフィールドがボトムフィールドより時間的に早いフレーム構造であることを表しており、Top_field_firstフラグの”０”は、ボトムフィールドがトップフィールドより時間的に早いフレーム構造であることを表している。

【００７０】図２０（Ａ）について、具体的に説明すると、最初にMPEGエンコード部１０３に入力されるFrame No１の符号化フレームの符号化画像データ種別は、Ｉピクチャであり、このＩピクチャの２フィールド（トップフィールドとボトムフィールド）は、トップフィールドをコピーしてリピートフィールドを作成することで、３フィールドに変換する必要があるので、対応するRepeat_first_fieldのフラグは”１”となり、Top_field_firstのフラグは”１”となる。

【００７１】Frame No２の符号化フレームの符号化画像データ種別は、Ｂピクチャであり、このＢピクチャには、リピートフィールドが生成する必要がないので、Repeat_first_fieldのフラグは”０”とされ、ボトムフィールドがトップフィールドより時間的に早いフレームであるため、Top_field_firstのフラグは”０”とされる。このときのTop_field_firstのフラグの値は、３－２プルダウン処理には関係しない。

【００７２】Frame No３の符号化フレームの符号化画像データ種別は、Ｂピクチャであり、Frame No３のＢピクチャでは、そのボトムフィールドをコピーしてリピートフィールドが作成され、符号化フレームが３フィールドに変換されている。従って、Repeat_first_fieldのフラグは”１”とされ、Top_field_firstのフラグは”０”とされる。

【００７３】Frame No４の符号化フレームの符号化画像データ種別は、Ｐピクチャであり、このＰピクチャに対しては、リピートフィールドが作成されておらず、Repeat_first_fieldのフラグは”０”とされ、Top_field_firstのフラグは１とされる。

【００７４】MPEGエンコード部１０３は、図２０（Ａ）に示すような３－２プルダウン処理が施された画像データが入力されてきたとき、内蔵するカウンタPTS_counterでフィールドの数をカウントし、その値PTS_counterを表示順序としてエンコードコントローラ１０２に出力する。カウンタPTS_counterは、０から１２７まで増加した後、再び０に戻るカウント動作を行う。従って、カウンタPTS_counterの値は、図２０（Ｂ）に示すように変化する。

【００７５】具体的に説明すると、最初に入力されるFrame No１のＩピクチャのPTS_counterの値は値”０”である。先頭から２番目に入力されるFrame No２のＢピクチャのPTS_counterの値は、Frame No１のＩピクチャの PTS_counterの値”０”に、Ｐピクチャのフィールド数３を加算した値”３”（＝０＋３）となる。

【００７６】先頭から３番目に入力されるFrame No３のＢピクチャのPTS_counterの値は、Frame No２のＢピクチャのPTS_counterの値”３”に、Ｂピクチャのフィールド数２を加算した値”５”（＝３＋２）となる。先頭から４番目に入力されるFrameNo４のＰピクチャのPTS_counterの値は、Frame No３のＢピクチャのPTS_counterの値”５”に、Ｂピクチャのフィールド数３を加算した値”８”（＝５＋３）となる。Frame No５のＢピクチャ以降のPTS_counterの値も同様に算出される。

【００７７】さらに、MPEGエンコード部１０３は、内蔵するカウンタDTS_counterでエンコードしたフレームを計数し、計数した結果をエンドコントローラ１０２に出力する。

【００７８】図２０（Ｃ）を参照して、具体的に説明すると、Frame No１のＩピクチャのDTS_counterの値１２５は、Frame No１のＩピクチャが表示される表示順序PTS_counter＝０を基準としたとき、１フレーム分の出現周期前に符号化される必要がある（図１４に対応させると、先頭のＩピクチャの符号化順序の値は”１”であり、表示順序の値は”２”であり、符号化順序の値は表示順序の値より１フレーム分早い必要がある）。つまり、Ｉピクチャが３つのフィールドを持っているため、DTS_counterの値は、０より３だけ前の値”１２５”（DTS_counterは2^7（＝１２８）のモジュロで表されるため、その値は０から１２７の間の値を循環する）となる。

【００７９】Frame No１のＩピクチャの次に符号化されるFrame No４のＰピクチャのDTS_counterの値は、Frame No１のＩピクチャのの値１２５にＩピクチャのフィールド数３を加えた値０（＝１２８＝１２５＋３）となる。

【００８０】Frame No４のＰピクチャの次に符号化される、Frame No２のＢピクチャのの値は、Ｂピクチャのの値は、PTS_counter＝DTS_counterであり、PTS_counterの値と同一とされ、その値は”３”となる。同様に、Frame No２のＢピクチャの次に符号化される、Frame No３のＢピクチャのの値も、PTS_counterの値と同一とされ、その値は”５”とされる。以下、Frame No７のＰピクチャ以降のDTS_counterの値も、同様に算出されるので、ここでは、その説明を省略

する。

【００８１】MPEGエンコード部１０３は、図２０に示すようなフラグRepeat_first_field, Top_field_first、並びにカウンタPTS_counter, DTS_counterをPOIとして、エンコードコントローラ１０２に出力する。

【００８２】ここで、図１１に示した、離れた場所にあるスタジオ４１のMPEGエンコーダ４２から、スタジオ７１のMPEGデコーダ７４に、SDTI-CPネットワーク５０，８０、TS MUX/DEMUX６１，６２、およびATMのネットワークを使用して、画像信号の伝送を行うシステムの、符号化順序と表示順序について、図２１を参照して説明する。

【００８３】MPEGエンコーダ４２内のMPEGビデオエンコーダ４２Ａは、MPEG符号化された画像データのエレメンタリストリーム（ES）を出力するとともに、POIを、そのエレメンタリストリーム（ES）中のuser dataに挿入する（図１５のエンコードコントローラ１０２がPOIのデータをuser dataに記述し、可変長符号化部１０４に出力して、多重化させる）。MPEGエンコーダ４２内のオーディオエンコーダ４２Ｂは、音声データを符号化してエレメンタリストリーム（ES）として出力する。SDTI-CPインタフェース４６は、MPEGビデオエンコーダ４２Ａからのエレメンタリストリーム（ES）（POIを含んだストリーム）と、オーディオエンコーダ４２Ｂからのエレメンタリストリーム（ES）を、SDTIフォーマットのエレメンタリストリームに変換して、SDTI-CPネットワーク５０に出力する。

【００８４】SDTI-CPインタフェース５１は、入力されたSDTIフォーマットのエレメンタリストリーム（ES）を、MPEG符号化のエレメンタリストリーム（ES）に変換し（図１８を参照してSDTI-CPインタフェース４６の動作として説明したように）、TSMUX/DEMUX６１に出力する。TS MUX/DEMUX６１は、エレメンタリストリーム（ES）に挿入されているPOIを参照して、PTS_counterの値をPTS（Presentation TimeStamp）に、また、DTS_counterの値をDTS（Decoding Time Stamp）に、それぞれ換算し、多重化処理を行い、１８８バイト単位のパケットから構成されるトランスポートストリーム（TS）を生成して出力する（従って、トランスポートストリーム（TS）には、PTS_counterとDTS_counterではなく、PTSとDTSが含まれる）。

【００８５】TS MUX/DEMUX６１は、エレメンタリストリーム（ES）に挿入されているPOIを解釈することにより、前述のバッファリング処理を行うことなく、直ちに多重化処理を行うことができ、TS MUX/DEMUX６１において、新たに遅延が発生することはない。また、POIはエレメンタリストリーム（ES）中に挿入されているため、TS MUX/DEMUX６１は、POIを後段に伝達するために、POIをビットストリームに含ませる処理を行わなくてもよい。

【００８６】図１１のTS MUX/DEMUX６２は、ATMなどの公衆網を介して入力されたトランスポートストリーム（TS）から画像データと音声データを分離して、MPEG符号化のエレメンタリストリーム（ES）に変換する。TS MUX/DEMUX６２はまた、PTS, DTSをPTS_counter, DTS_counterに換算して、エレメンタリストリームのuser dataに挿入し、SDTI-CPインタフェース８１に出力する。SDTI-CPインタフェース８１は、MPEG符号化されたエレメンタリストリーム（ES）をSDTIフォーマットのエレメンタリストリーム（ES）に変換し、スタジオ７１のSDTI-CPネットワーク８０に出力する。MPEGデコーダ７４（MPEGデコーダ４４と同一の構成）は、SDTI-CPインタフェース７８を介して、伝送されてきた画像データと音声データのエレメンタリストリーム（ES）を受信し、復号する。

【００８７】MPEGデコーダ７４は、TS MUX/DEMUX６１と同様に、エレメンタリストリーム（ES）に挿入されているPOIを解釈して、前述のバッファリング処理を行うことなく、直ちに復号する（図１６のデコードコントローラ１１３が出力するPOIに基づき、MPEGデコード部１１４が画像データを復号する）ことができ、MPEGデコーダ７４において、新たに遅延が発生することはない。つまり、MPEGビデオエンコーダ４２Ａが、MPEG符号化されたエレメンタリストリーム（ES）とともに、POIをエレメンタリストリームに挿入して出力することにより、後段のTS MUX/DEMUX６１と、MPEGデコーダ７４は、POIを解釈して多重化処理、または復号処理を直ちに行うことができ、システム全体としての遅延を、MPEGエンコーダ４２で生ずるＢピクチャの枚数（＝２）＋１枚＝３の遅延のみに抑えることができる。すなわち、このような符号化、多重化、および復号を含むシステムにおいて、理論的に最も小さい遅延とすることができる。

【００８８】また、図２１のシステム構成において、図１８に示したSDTI-CPインタフェース４６を用いることにより、SDTI-CPネットワーク５０を使用して画像信号を伝送するとき、スタジオ４１の内部では、ビットストリームを、編集が容易で、短距離伝送に適したエレメンタリストリーム（ES）の形態で伝送することが可能となり、離れたスタジオ間でATMなどの公衆網を使用して画像信号を伝送するとき、ビットストリームを、長距離伝送に適したトランスポートストリーム（TS）の形態で伝送することが可能となる。

【００８９】以上においては、POIをエレメンタリストリームに挿入するようにしたが、TSMUX/DEMUX６１とMPEGエンコーダ４２の距離が近いような場合には、図２２に示すように、POIをMPEGエンコーダ４２から、TS MUX/DEMUX６１に直接供給するようにしてもよい。

【００９０】しかしながら、このようにすると、エレメンタリストリームを伝送するSDTI-CPネットワーク５０以外の配線処理が必要となる。

【００９１】以上に述べたように、エンコーダの有する情報を、エレメンタリストリーム（ES）中のuser dataに記述して、多重化装置またはデコーダに出力することにより、エンコーダのみが有していた情報（Coding Phase（V-PhaseとH-Phase）、Field_ID、符号化順序DTS_counter、および表示順序PTS_counter）を、エンコーダより後段の多重化装置、デコーダに供給することができる。

【００９２】次に、図２３乃至図３６を参照してビットストリームのシンタックスについて説明する。

【００９３】図２３は、MPEGのビデオストリームのシンタックスを表わした図である。MPEGエンコーダ４２は、この図２３に示されたシンタックスに従った符号化エレメンタリストリームを生成する。以下に説明するシンタックスにおいて、関数や条件文は細活字で表わされ、データエレメントは、太活字で表されている。データ項目は、その名称、ビット長およびそのタイプ・伝送順序を示すニーモニック（Mnemonic）で記述されている。

【００９４】まず、この図２３に示されているシンタックスにおいて使用されている関数について説明する。実際には、この図２３に示されているシンタックスは、MPEGデコーダ４４側において、伝送されてきた符号化ビットストリームから所定の意味のあるデータエレメントを抽出するために使用されるシンタックスである。MPEGエンコーダ４２側において使用されるシンタックスは、図２３に示されたシンタックスからif文やwhile文等の条件文を省略したシンタックスである。

【００９５】video_sequence()において最初に記述されているnext_start_code()関数は、ビットストリーム中に記述されているスタートコードを探すための関数である。この図２３に示されたシンタックスに従って生成された符号化ストリームには、まず最初に、sequence_header()関数とsequence_extension()関数によって定義されたデータエレメントが記述されている。このsequence_header()関数は、MPEGビットストリームのシーケンスレイヤのヘッダデータを定義するための関数であって、sequence_extension()関数は、MPEGビットストリームのシーケンスレイヤの拡張データを定義するための関数である。

【００９６】sequence_extension()関数の次に配置されている do{}while構文は、while文によって定義されている条件が真である間、do文の{}内の関数に基いて記述されたデータエレメントが符号化データストリーム中に記述されていることを示す構文である。このwhile文に使用されているnextbits()関数は、ビットストリーム中に記述されているビット又はビット列と、参照されるデータエレメントとを比較するための関数である。この図２３に示されたシンタックスの例では、nextbits()関数は、ビットストリーム中のビット列とビデオシーケンスの終わりを示すsequence_end_codeとを比較し、ビットストリーム中のビット列とsequence_end_codeとが一致しないときに、このwhile文の条件が真となる。従って、sequence_extension()関数の次に配置されている do{}while構文は、ビットストリーム中に、ビデオシーケンスの終わりを示すsequence_end_codeが現れない間、do文中の関数によって定義されたデータエレメントが符号化ビットストリーム中に記述されていることを示している。

【００９７】符号化ビットストリームにおいて、sequence_extension()関数によって定義された各データエレメントの次には、extension_and_user_data(0)関数によって定義されたデータエレメントが記述されている。このextension_and_user_data(0)関数は、MPEGビットストリームのシーケンスレイヤにおける拡張データとユーザデータを定義するための関数である。

【００９８】このextension_and_user_data(0)関数の次に配置されている do{}while構文は、while文によって定義されている条件が真である間、do文の{}内の関数に基いて記述されたデータエレメントが、ビットストリームに記述されていることを示す関数である。このwhile文において使用されているnextbits()関数は、ビットストリーム中に現れるビット又はビット列と、picture_start_code又はgroup_start_codeとの一致を判断するための関数であるって、ビットストリーム中に現れるビット又はビット列と、picture_start_code又はgroup_start_codeとが一致する場合には、while文によって定義された条件が真となる。よって、このdo{}while構文は、符号化ビットストリーム中において、picture_start_code又はgroup_start_codeが現れた場合には、そのスタートコードの次に、do文中の関数によって定義されたデータエレメントのコードが記述されていることを示している。

【００９９】このdo文の最初に記述されているif文は、符号化ビットストリーム中にgroup_start_codeが現れた場合、という条件を示しいる。このif文による条件は真である場合には、符号化ビットストリーム中には、このgroup_start_codeの次にgroup_of_picture_header()関数およびextension_and_user_data(1)関数によって定義されているデータエレメントが順に記述されている。

【０１００】このgroup_of_picture_header()関数は、MPEG符号化ビットストリームのGOPレイヤのヘッダデータを定義するための関数であって、extension_and_user_data(1)関数は、MPEG符号化ビットストリームのGOPレイヤの拡張データおよびユーザデータを定義するための関数である。

【０１０１】さらに、この符号化ビットストリームにおいて、group_of_picture_header()関数およびextension_and_user_data(1)関数によって定義されているデータエレメントの次には、picture_header()関数とpicture_coding_extension()関数によって定義されたデータエレ

メントが記述されている。もちろん、先に説明したif文の条件が真とならない場合には、 group_of_picture_header()関数およびextension_and_user_data(1)関数によって定義されているデータエレメントは記述されていないので、 extension_and_user_data(0)関数によって定義されているデータエレメントの次に、picture_header()関数、picture_coding_extension()関数およびextension_and_user_data(2)関数によって定義されたデータエレメントが記述されている。

【０１０２】このpicture_header()関数は、MPEG符号化ビットストリームのピクチャレイヤのヘッダデータを定義するための関数であって、picture_coding_extension()関数は、MPEG符号化ビットストリームのピクチャレイヤの第１の拡張データを定義するための関数である。extension_and_user_data(2)関数は、MPEG符号化ビットストリームのピクチャレイヤの拡張データおよびユーザデータを定義するための関数である。このextension_and_user_data(2)関数によって定義されるユーザデータは、ピクチャレイヤに記述されているデータであって、各ピクチャ毎に記述することのできるデータである。

【０１０３】符号化ビットストリームにおいて、ピクチャレイヤのユーザデータの次には、picture_data()関数によって定義されるデータエレメントが記述されている。このpicture_data()関数は、スライスレイヤおよびマクロブロックレイヤに関するデータエレメントを記述するための関数である。

【０１０４】このpicture_data()関数の次に記述されているwhile文は、このwhile文によって定義されている条件が真である間、次のif文の条件判断を行うための関数である。このwhile文において使用されているnextbits()関数は、符号化ビットストリーム中に、picture_start_code又はgroup_start_codeが記述されているか否かを判断するための関数であって、ビットストリーム中にpicture_start_code又はgroup_start_codeが記述されている場合には、このwhile文によって定義された条件が真となる。

【０１０５】次のif文は、符号化ビットストリーム中にsequence_end_code が記述されているか否かを判断するための条件文であって、sequence_end_code が記述されていないのであれば、sequence_header()関数とsequence_extension()関数とによって定義されたデータエレメントが記述されていることを示している。sequence_end_codeは符号化ビデオストリームのシーケンスの終わりを示すコードであるので、符号化ストリームが終了しない限り、符号化ストリーム中にはsequence_header()関数とsequence_extension()関数とによって定義されたデータエレメントが記述されている。

【０１０６】このsequence_header()関数とsequence_extension()関数によって記述されたデータエレメントは、ビデオストリームのシーケンスの先頭に記述されたsequence_header()関数とsequence_extension()関数によって記述されたデータエレメントと全く同じである。このように同じデータをストリーム中に記述する理由は、ビットストリーム受信装置側でデータストリームの途中（例えばピクチャレイヤに対応するビットストリーム部分）から受信が開始された場合に、シーケンスレイヤのデータを受信できなくなり、ストリームをデコード出来なくなることを防止するためである。

【０１０７】この最後のsequence_header()関数とsequence_extension()関数とによって定義されたデータエレメントの次、つまり、データストリームの最後には、シーケンスの終わりを示す２ビットのsequence_end_codeが記述されている。

【０１０８】以下に、sequence_header()関数、sequence_extension()関数、extension_and_user_data(0)関数、group_of_picture_header()関数、picture_header()関数、picture_coding_extension()関数、およびpicture_data()関数について詳細に説明する。

【０１０９】図２４は、sequence_header()関数のシンタックスを説明するための図である。このsequence_header()関数によって定義されたデータエレメントは、sequence_header_code、horizontal_size_value、vertical_size_value、aspect_ratio_information、frame_rate_code、bit_rate_value、marker_bit、vbv_buffer_size_value、constrained_parameter_flag、load_intra_quantizer_matrix、intra_quantizer_matrix[64]、load_non_intra_quantizer_matrix、およびnon_intra_quantizer_matrix等である。

【０１１０】sequence_header_codeは、シーケンスレイヤのスタート同期コードを表すデータである。horizontal_size_valueは、画像の水平方向の画素数の下位１２ビットから成るデータである。vertical_size_valueは、画像の縦のライン数の下位１２ビットからなるデータである。aspect_ratio_informationは、画素のアスペクト比（縦横比）または表示画面アスペクト比を表すデータである。frame_rate_codeは、画像の表示周期を表すデータである。bit_rate_valueは、発生ビット量に対する制限のためのビット・レートの下位１８ビット（４００bsp単位で切り上げる）データである。marker_bitは、スタートコードエミュレーションを防止するために挿入されるビットデータである。vbv_buffer_size_valueは、発生符号量制御用の仮想バッファ（ビデオバッファベリファイヤー）の大きさを決める値の下位１０ビットデータである。constrained_parameter_flagは、各パラメータが制限以内であることを示すデータである。load_intra_quantizer_matrixは、イントラMB用量子化マトリックス・データの存在を示すデータである。intra_quantizer_matrix[64]は、イントラMB用量子化マトリックスの値を示すデータである。load_non_intra_quantizer_matrixは、非イントラMB用量子化マトリックス・デ

ータの存在を示すデータである。non_intra_quantizer_matrixは、非イントラMB用量子化マトリックスの値を表すデータである。

【０１１１】図２５はsequence_extension()関数のシンタックスを説明するための図である。このsequence_extension()関数によって定義されたデータエレメントとは、extension_start_code、extension_start_code_identifier、profile_and_level_indication、progressive_sequence、chroma_format、horizontal_size_extension、vertical_size_extension、bit_rate_extension、vbv_buffer_size_extension、low_delay、frame_rate_extension_n 、および frame_rate_extension_d等のデータエレメントである。

【０１１２】extension_start_codeは、エクステンションデータのスタート同期コードを表すデータである。extension_start_code_identifierは、どの拡張データが送られるかを示すデータである。profile_and_level_indicationは、ビデオデータのプロファイルとレベルを指定するためのデータである。progressive_sequenceは、ビデオデータが順次走査であることを示すデータである。chroma_formatは、ビデオデータの色差フォーマットを指定するためのデータである。horizontal_size_extensionは、シーケンスヘッダのhoriznta_size_valueに加える上位２ビットのデータである。vertical_size_extensionは、シーケンスヘッダのvertical_size_value加える上位２ビットのデータである。bit_rate_extensionは、シーケンスヘッダのbit_rate_valueに加える上位１２ビットのデータである。vbv_buffer_size_extensionは、シーケンスヘッダのvbv_buffer_size_valueに加える上位８ビットのデータである。low_delayは、Ｂピクチャを含まないことを示すデータである。frame_rate_extension_nは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを得るためのデータである。frame_rate_extension_dは、シーケンスヘッダのframe_rate_codeと組み合わせてフレームレートを得るためのデータである。

【０１１３】図２６は、extension_and_user_data(i)関数のシンタックスを説明するための図である。このextension_and_user_data(i)関数は、「i」が１以外のときは、extension_data()関数によって定義されるデータエレメントは記述せずに、user_data()関数によって定義されるデータエレメントのみを記述する。よって、extension_and_user_data(0)関数は、user_data()関数によって定義されるデータエレメントのみを記述する。

【０１１４】まず、図２６に示されているシンタックスにおいて使用されている関数について説明する。nextbits()関数は、ビットストリーム中に現れるビットまたはビット列と、次に復号されるデータエレメントとを比較するための関数である。

【０１１５】user_data()関数は、図２７に示すように、user_data_start_code, V-phase () 関数,H-phase()関数, Time_code()関数, Picture-order()関数, Ancillary_data()関数, history_data()関数, およびuser_dataのデータエレメントを記述するための関数である。

【０１１６】user_data_start_codeは、MPEG方式のビットストリームのピクチャレイヤのユーザデータエリアの開始を示すためのスタートコードである。このuser_data_start_codeの次に記述されているif文は、user_data(ｉ)関数のiが”０”のとき、次に記述されているwhile構文を実行する。このwhile構文は、ビットストリーム中に、２３個の”０”とそれに続く”１”から構成される２４ビットのデータが現れない限り真となる。

【０１１７】この２３個の”０”とそれに続く”１”から構成される２４ビットのデータは、すべてのスタートコードの先頭に付与されるデータであって、すべてのスタートコードは、この２４ビットの後ろに設けられることによって、nextbits()関数は、ビットストリーム中において、各スタートコードの位置を見つけることができる。

【０１１８】while構文が真のとき、その次に記述されているif文のnextbits()関数は、V-Phaseを示すビット列（Data_ID）を検出すると、そのビット列（Data_ID）の次ビットからV-Phase()関数で示されるV-Phaseのデータエレメントが記述されていることを知る。次のElse if文のnextbits()関数は、H-Phaseを示すビット列（Data_ID）を検出すると、そのビット列（Data_ID）の次ビットからH-Phase()関数で示されるH-Phaseのデータエレメントが記述されていることを知る。

【０１１９】ここで、図２８に示すように、V-PhaseのData_IDは、”０１”を表すビット列であり、H-PhaseのData_IDは、”０２”を表すビット列である。

【０１２０】ビットストリームに記述されるV-Phase()関数のシンタックスについて、図２９を参照して説明する。まず、Data_IDは、前述したように、そのData_IDの次のビット列のデータエレメントがV-Phaseであることを表す８ビットのデータであり、図２８で示した値”０１”である。V-Phaseは、画像信号のフレームにおいて、符号化される最初のラインを示す１６ビットのデータである。

【０１２１】ビットストリームに記述されるH-Phase()関数のシンタックスについて、図３０を参照して説明する。まず、Data_IDは、前述したように、そのData_IDの次のビット列のデータエレメントがH-Phaseであることを表す８ビットのデータであり、図２８で示した値”０２”である。H-Phaseは、画像信号フレームにおいて、符号化される最初のサンプルを示す８ビットのデータである。

【０１２２】図２７に戻って、次のElse if文は、user_data(ｉ)関数のiが２のとき、次に記述されているwhile構文を実行する。while構文の内容は前述した場合と同

様であるので、ここではその説明を省略する。

【０１２３】while構文が真のとき、次のif文において、nextbits()関数は、Time code1を示すビット列を検出するか、または、Time code２を示すビット列を検出すると、そのビット列の次ビットからTime_code()関数で示されるTime codeのデータエレメントが記述されていることを知る。

【０１２４】Time code1のData_IDは、図２８に示すように、”０３”を表すビット列であり、Time code1のデータは、画像の垂直ブランキング期間に挿入されたタイムコードを示す、VITC（Vertical Interval Time Code）である。Time code２のData_IDは、図２８に示すように、”０４”を表すビット列であり、Time code２のデータは、記録媒体のタイムコードトラックに記録されたタイムコードを示す、LTC（Longitudinal Time Code）である。

【０１２５】次に、Else if文において、nextbits()関数は、Picture Orderを示すビット列を検出すると、そのビット列の次ビットからPicture_Order()関数で示されるPicture Orderのデータエレメントが記述されていることを知る。ここで、Picture_Order()関数のData_IDは、図２８に示すように、”０５”を表すビット列である。

【０１２６】実際に、エンコーダでエレメンタリストリーム（ES）に挿入するPicture_Order()関数のシンタックスを、図３１を参照して説明する。まず、Data_IDは前述したように、そのData_ID以降のデータがPOIのデータであることを示す８ビットのデータであり、その値は”０５”である。DTS_presenceは、符号化順序DTS_counterの有無を表す１ビットのデータである。例えば、ＢピクチャのようにDTS_counter＝PTS_counterとなる場合、表示順序PTS_counterのみが存在し、DTS_presenceのビットは”０”となる。逆に、ＰピクチャおよびＩピクチャの場合、符号化順序DTS_counterと表示順序PTS_counterは同一ではないので、表示順序PTS_counterと符号化順序DTS_counterの双方が存在し、DTS_presenceのビットは１となる。

【０１２７】PTS_counterは、エンコーダに符号化フレーム中の１フィールドが入力されるごとにカウントアップを行う、表示順序を表す７ビットのデータである。この７ビットのデータは、０から１２７までの値をとるモジュロである。if文以降は、DTS_presenceのビットが１のとき、すなわち、ＰピクチャおよびＩピクチャのとき、DTS_counterのカウントアップが実行される。

【０１２８】Marker_bitsは、user dataの記述されたビット列が、偶然に前述したスタートコードと一致し、画像破錠を引き起こす可能正が高い、スタートコードエミュレーションを防止するために、１６ビットごとに挿入されるビットである。

【０１２９】DTS_counterは、エンコーダで、１フィールド分の符号化画像データが符号化されるごとにカウントアップを行う、符号化順序を表す７ビットのデータである。この７ビットのデータは、０から１２７までの値をとるモジュロである。

【０１３０】前述したように、表示順序PTS_counterは、フィールド単位で付番されるために、例えば、符号化画像データを２４Hzから３０Hzのフレームレートに変換して符号化する場合、３－２プルダウン処理を行った後に、付番する必要がある。

【０１３１】図２７に戻って、その次に記述されているwhile構文も、内容は前述した場合と同様であるので、ここではその説明を省略する。while構文が真のとき、次のif文において、nextbits()関数は、Ancillary dataを示すビット列を検出すると、そのビット列の次ビットからAncillary_data()関数で示されるAncillary dataのデータエレメントが記述されていることを知る。Ancillary_data()関数のData_IDは、図２８に示すように、”０７”を表すビット列である。

【０１３２】この補助データに識別子を付加するancillary dataのシンタックスを図３２を参照して説明する。Ancillary_data()関数はピクチャ層のuser dataとして伝送され、データとしてはField識別子（Field_ID）、ラインの番号（Line_number）および補如データ（ancillary data）が挿入される。

【０１３３】Data_IDは、user data領域において、ancillary dataであることを示す８ビットのデータであり、その値は図２８に示したように”０７”である。

【０１３４】Field_IDは２ビットのデータであり、progressive_sequence flag（図２５）の値が”０”のとき、符号化フレーム内のフィールドごとにField_IDが付加される。repeat_first_fieldに”０”が設定されているとき、この符号化フレームにはフィールドが２枚存在し、Field_IDは、図１９に示したように、最初のフィールドに”０”、およびその次のフィールドに”１”が設定され、repeat_first_fieldに”１”が設定されているとき、この符号化フレームにはフィールドが３枚存在し、Field_IDとしては、最初のフィールドに”０”が設定され、それ以降のフィールドに”１”，”２”が設定される。

【０１３５】Field_IDは、progressive_sequence flagの値が”１”のとき、符号化フレームごとに付加される。Field_IDには、repeat_first_fieldとTop_field_firstにともに”０”が設定されているとき、その符号化フレームは１枚のprogressiveframeが存在するので、値”０”が設定され、repeat_first_fieldに値”１”およびTop_field_firstに値”０”が設定されているとき、その符号化フレームは２枚のprogressive frameが存在するので、値”０”，”１”が設定され、repeat_first_fieldとTop_field_firstにともに”１”が設定されているとき、その符号化フレームは３枚のprogressiv

e frameが存在するので、値”０”乃至”２”が設定される。

【０１３６】Line_numberは、１４ビットのデータであり、各フレームにおける補助データが記述されている、ITU-R BT.656-3,SMPTE274M,SMPTE293M,SMPTE296Mで規定されたライン番号を示す。

【０１３７】Ancillary_data_lengthは、１６ビットのデータであり、ancillary_data_payloadのデータ長を示す。Ancillary_data_payloadは、２２ビットのデータからなる補助データの内容を表しており、Ancillary_data_payloadのAncillary_data_lengthの値がｊの値（初期値０）より大きいとき、値ｊ（Ancillary_data_lengthのデータ長）を１だけインクリメントして、そのｊの値のビット列目から記述される。

【０１３８】次のWhile構文は、bytealigned()関数のためのシンタックスを表しており、次のデータがbytealigned()関数でないとき（While構文が真のとき）、Zero_bit（１ビットのデータ”０”）を記述する。

【０１３９】図２７に戻って、次のElse if文において、nextbits()関数は、History dataを示すビット列を検出すると、そのビット列の次ビットからHistory_data() 関数で示されるHistory dataのデータエレメントが記述されていることを知る。History_data () 関数のData_IDは、図２８に示すように、”０８”を表すビット列であり、Data_IDが”０８”で示されるデータは、符号化パラメータの履歴情報を含むHistory dataを表している。

【０１４０】最後のif文において、nextbits()関数は、user dataを示すビット列を検出すると、そのビット列の次ビットからuser_data()関数で示されるuser_dataのデータエレメントが記述されていることを知る。

【０１４１】図２７のnextbits()関数が、それぞれのデータエレメントが記述されていることを知るビット列は、図２８に示すData_IDとして記述されている。ただし、Data_IDとして”００”を使用することは禁止されている。Data_IDが”８０”で示されるデータは、制御フラグを表しており、Data_IDが”ＦＦ”で示されるデータは、user dataを表している。

【０１４２】図３３は、group_of_picture_header()関数のシンタックスを説明するための図である。このgroup_of_picture_header()関数によって定義されたデータエレメントは、group_start_code、time_code、closed_gop、およびbroken_linkから構成される。

【０１４３】group_start_codeは、GOPレイヤの開始同期コードを示すデータである。time_codeは、GOPの先頭ピクチャのシーケンスの先頭からの時間を示すタイムコードである。closed_gopは、GOP内の画像が他のGOPから独立再生可能なことを示すフラグデータである。broken_linkは、編集などのためにGOP内の先頭のＢピクチャが正確に再生できないことを示すフラグデータである。

【０１４４】extension_and_user_data(1)関数は、 extension_and_user_data(0)関数と同じように、user_data()関数によって定義されるデータエレメントのみを記述するための関数である。

【０１４５】次に、図３４乃至図３６を参照して、符号化ストリームのピクチャレイヤに関するデータエレメントを記述するためのpicture_headr()関数、picture_coding_extension()関数、およびpicture_data()について説明する。

【０１４６】図３４はpicture_headr()関数のシンタックスを説明するための図である。このpicture_headr()関数によって定義されたデータエレメントは、picture_start_code、temporal_reference、picture_coding_type、vbv_delay、full_pel_forward_vector、forward_f_code、full_pel_backward_vector、backward_f_code、extra_bit_picture、およびextra_information_pictureである。

【０１４７】具体的には、picture_start_codeは、ピクチャレイヤの開始同期コードを表すデータである。temporal_referenceは、ピクチャの表示順を示す番号で、GOPの先頭でリセットされるデータである。picture_coding_typeは、ピクチャタイプを示すデータである。

【０１４８】vbv_delayは、VBVバッファの初期状態を示すデータであって、各ピクチャ毎に設定されている。送信側システムから受信側システムに伝送された符号化エレメンタリストリームのピクチャは、受信側システムに設けられたＶＢＶバッファにバッファリングされ、DTS（Decoding Time Stamp）によって指定された時刻に、このVBVバッファから引き出され（読み出され）、デコーダに供給される。vbv_delayによって定義される時間は、復号化対象のピクチャがVBVバッファにバッファリングされ始めてから、符号化対象のピクチャがVBVバッファから引き出されるまでの時間、つまりDTSによって指定された時刻までの時間を意味する。このピクチャヘッダに格納されたvbv_delayを使用することによって、VBVバッファのデータ占有量が不連続にならないシームレスなスプライシングが実現できる。

【０１４９】full_pel_forward_vectorは、順方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。forward_f_codeは、順方向動きベクトル探索範囲を表すデータである。full_pel_backward_vectorは、逆方向動きベクトルの精度が整数単位か半画素単位かを示すデータである。backward_f_codeは、逆方向動きベクトル探索範囲を表すデータである。extra_bit_pictureは、後続する追加情報の存在を示すフラグである。このextra_bit_pictureが「１」の場合には、次にextra_information_pictureが存在し、extra_bit_pictureが「０」の場合には、これに続くデータが無いことを示している。extra_information_pictureは、規格において予約された情報である。

【０１５０】図３５は、picture_coding_extension()関数のシンタックスを説明するための図である。このpicture_coding_extension()関数によって定義されたデータエレメントは、extension_start_code、extension_start_code_identifier、f_code[0][0]、f_code[0][1]、f_code[1][0]、f_code[1][1]、intra_dc_precision、picture_structure、top_field_first、frame_predictive_frame_dct、concealment_motion_vectors、q_scale_type、intra_vlc_format、alternate_scan、repeat_first_field、chroma_420_type、progressive_frame、composite_display_flag、v_axis、field_sequence、sub_carrier、burst_amplitude、およびsub_carrier_phaseから構成される。

【０１５１】extension_start_codeは、ピクチャレイヤのエクステンションデータのスタートを示す開始コードである。extension_start_code_identifierは、どの拡張データが送られるかを示すコードである。f_code[0][0]は、フォアード方向の水平動きベクトル探索範囲を表すデータである。f_code[0][1]は、フォアード方向の垂直動きベクトル探索範囲を表すデータである。f_code[1][0]は、バックワード方向の水平動きベクトル探索範囲を表すデータである。f_code[1][1]は、バックワード方向の垂直動きベクトル探索範囲を表すデータである。intra_dc_precisionは、DC係数の精度を表すデータである。picture_structureは、フレームストラクチャかフィールドストラクチャかを示すデータである。これは、フィールドストラクチャの場合は、上位フィールドか下位フィールドかもあわせて示す。

【０１５２】top_field_firstは、フレームストラクチャの場合、最初のフィールドがトップフィールドであるのか、ボトムフィールドであるのかを示すフラグである。frame_predictive_frame_dctは、フレーム・ストラクチャの場合、フレーム・モードDCTの予測がフレーム・モードだけであることを示すデータである。concealment_motion_vectorsは、イントラマクロブロックに伝送エラーを隠蔽するための動きベクトルがついていることを示すデータである。q_scale_typeは、線形量子化スケールを利用するか、非線形量子化スケールを利用するかを示すデータである。intra_vlc_formatは、イントラマクロブロックに、別の２次元VLC（可変長符号）を使うかどうかを示すデータである。alternate_scanは、ジグザグスキャンを使うか、オルタネート・スキャンを使うかの選択を表すデータである。

【０１５３】repeat_first_fieldは、復号化時にリピートフィールドを生成するか否かを示すフラグであって、復号化時の処理において、repeat_first_field が「１」の場合にはリピートフィールドを生成し、repeat_first_fieldが「０」の場合にはリピートフィールドを生成しないという処理が行われる。

【０１５４】chroma_420_typeは、信号フォーマットが４：２：０の場合、次のprogressive_frame と同じ値、そうでない場合は０を表すデータである。progressive_frameは、そのピクチャが、順次走査できているかどうかを示すデータである。composite_display_flagは、ソース信号がコンポジット信号であったかどうかを示すデータである。v_axisは、ソース信号が、PALの場合に使われるデータである。field_sequenceは、ソース信号が、PALの場合に使われるデータである。sub_carrierは、ソース信号が、PALの場合に使われるデータである。burst_amplitudeは、ソース信号が、PALの場合に使われるデータである。sub_carrier_phaseは、ソース信号が、PALの場合に使われるデータである。

【０１５５】図３６は、picture_data()関数のシンタックスを説明するための図である。このpicture_data()関数によって定義されるデータエレメントは、slice()関数によって定義されるデータエレメントである。但し、ビットストリーム中に、slice()関数のスタートコードを示すslice_start_codeが存在しない場合には、このslice()関数によって定義されるデータエレメントはビットストリーム中に記述されていない。

【０１５６】slice()関数は、スライスレイヤに関するデータエレメントを記述するための関数であって、具体的には、slice_start_code、slice_quantiser_scale_code、intra_slice_flag、intra_slice、reserved_bits、extra_bit_slice、extra_information_slice、およびextra_bit_slice 等のデータエレメントと、macroblock()関数によって定義されるデータエレメントを記述するための関数である。

【０１５７】slice_start_codeは、slice()関数によって定義されるデータエレメントのスタートを示すスタートコードである。slice_quantiser_scale_codeは、このスライスレイヤに存在するマクロブロックに対して設定された量子化ステップサイズを示すデータである。しかし、各マクロブロック毎に、quantiser_scale_codeが設定されている場合には、各マクロブロックに対して設定されたmacroblock_quantiser_scale_codeのデータが優先して使用される。intra_slice_flagは、ビットストリーム中にintra_sliceおよびreserved_bitsが存在するか否かを示すフラグである。intra_sliceは、スライスレイヤ中にノンイントラマクロブロックが存在するか否かを示すデータである。スライスレイヤにおけるマクロブロックのいずれかがノンイントラマクロブロックである場合には、intra_sliceは「０」となり、スライスレイヤにおけるマクロブロックの全てがノンイントラマクロブロックである場合には、intra_sliceは「１」となる。reserved_bitsは、７ビットのデータであって「０」の値を取る。extra_bit_sliceは、符号化ストリームとして追加の情報が存在することを示すフラグであって、次にextra_information_sliceが存在する場合には「１」に設定される。追加の情報が存在しない場合に

は「０」に設定される。

【０１５８】macroblock()関数は、マクロブロックレイヤに関するデータエレメントを記述するための関数であって、具体的には、macroblock_escape、macroblock_address_increment、およびmacroblock_quantiser_scale_code等のデータエレメントと、macroblock_modes()関数、および macroblock_vecters(s)関数によって定義されたデータエレメントを記述するための関数である。

【０１５９】macroblock_escapeは、参照マクロブロックと前のマクロブロックとの水平方向の差が３４以上であるか否かを示す固定ビット列である。参照マクロブロックと前のマクロブロックとの水平方向の差が３４以上の場合には、macroblock_address_incrementの値に３３をプラスする。macroblock_address_incrementは、参照マクロブロックと前のマクロブロックとの水平方向の差を示すデータである。もし、このmacroblock_address_incrementの前にmacroblock_escapeが１つ存在するのであれば、このmacroblock_address_incrementの値に３３をプラスした値が、実際の参照マクロブロックと前のマクロブロックとの水平方向の差分を示すデータとなる。macroblock_quantiser_scale_codeは、各マクロブロック毎に設定された量子化ステップサイズである。各スライスレイヤには、スライスレイヤの量子化ステップサイズを示すslice_quantiser_scale_codeが設定されているが、参照マクロブロックに対してmacroblock_quantiser_scale_codeが設定されている場合には、この量子化ステップサイズを選択する。

【０１６０】図３７は、MPEG符号化ストリームのデータ構造を示す説明図である。この図に示したように、ビデオエレメンタリストリームのデータ構造は、少なくともシーケンスレイヤ、GOPレイヤ、およびピクチャレイヤを含んでいる。

【０１６１】シーケンスレイヤは、next_start_code()関数２０１、sequence_header()関数２０２、extension_start_code()２０３、sequence_extension()関数２０４、extension_and_user_data(0)関数２０５によって定義されるデータエレメントから構成されている。GOPレイヤは、group_start_code２０６、group_of_picture_header()関数２０７、extension_and_user_data(1)関数２０８によって定義されるデータエレメントから構成されている。ピクチャレイヤは、picture_header()関数２０９、picture_coding_extension()関数２１０、extension_and_user_data(2)関数２１１、picture_data()関数２１２によって定義されるデータエレメントを含んでいる。ビデオシーケンスの最後には、sequence_end_code２１３が記述されている。

【０１６２】extension_and_user_data(2)関数２１１は、既に図２６において説明したシンタックスからも理解できるように、user_data_start_code２１４、user_data()関数２１５、next_start_code２１６によって定義されるデータエレメントを含んでいる。

【０１６３】user_data()関数２１５は、既に図２７において説明したシンタックスからも理解できるように、time_code()関数２１７とuser_data２１８によって定義されるデータエレメントを含んでいる。

【０１６４】なお、本明細書において、システムとは、複数の装置により構成される装置全体を表すものとする。

【０１６５】また、本明細書中において、上記処理を実行するコンピュータプログラムをユーザに提供する提供媒体には、磁気ディスク、CD-ROMなどの情報記録媒体の他、インターネット、デジタル衛星などのネットワークによる伝送媒体も含まれる。

【０１６６】

【発明の効果】以上の如く、請求項１に記載のデジタル信号伝送装置、請求項４に記載のデジタル信号伝送方法、および請求項５に記載の提供媒体によれば、第１のビットストリームから順序情報を抽出し、抽出した順序情報に基づいてタイムスタンプを生成し、第２のビットストリームに挿入するようにしたので、所定の場所における画像データの処理を迅速に行うことができるようになるとともに、他の場所においても、画像データを迅速に処理することが可能となる。

【図面の簡単な説明】

【図１】Coding Phaseを説明する図である。

【図２】補助データを説明する図である。

【図３】エレメンタリストリームを伝送するシステムの構成を示す図である。

【図４】補助データを伝送するシステムの構成を示す図である。

【図５】MPEG方式における予測構造と符号化構造を説明する図である。

【図６】MPEGエンコーダで発生する遅延について説明する図である。

【図７】スタジオの内部の構成を示すブロック図である。

【図８】図７のスタジオの内部で伝送されるデータを説明する図である。

【図９】図７のシステムにおいてデータを伝送するシステムの構成を示す図である。

【図１０】３－２プルダウン処理された画像の補助データを説明する図である。

【図１１】２つのスタジオの間でビットストリームを伝送するときのシステム構成を表すブロック図である。

【図１２】図１１のシステムで伝送されるストリームの構成を表す図である。

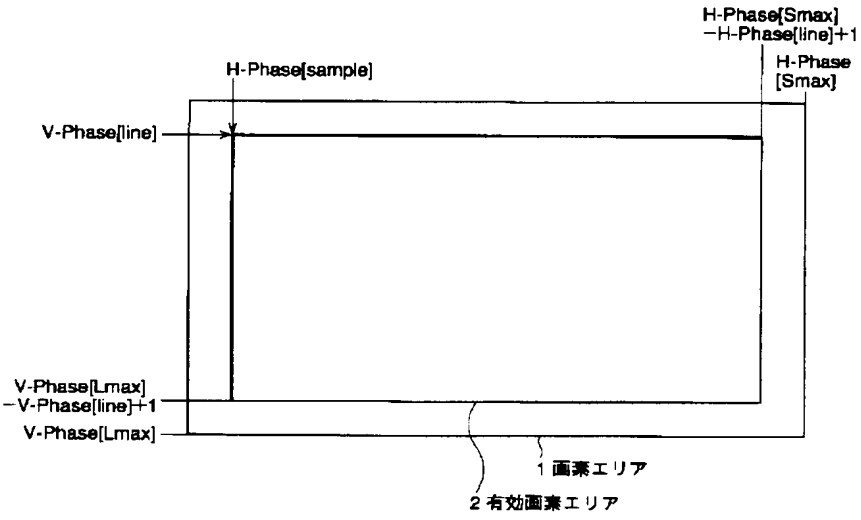
【図１３】図１１のシステムにおいて、トランスポートストリームを伝送する構成を表すブロック図である。

【図１４】符号化処理の遅延を説明する図である。

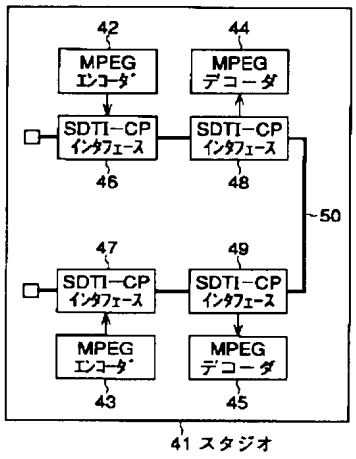
【図１５】本発明を適用したMPEGエンコーダの構成を表

すブロック図である。
【図１６】本発明を適用したMPEGデコーダの構成を表すブロック図である。
【図１７】CPIを伝送する場合のシステムの構成を表す図である。
【図１８】図１１のSDTI-CPインタフェース４６の構成を表すブロック図である。
【図１９】補助データの伝送を説明する図である。
【図２０】３－２プルダウン処理におけるPTS_counterとDTS_counterを説明する図である。
【図２１】POIを伝送する場合のシステムの構成を表す図である。
【図２２】POIを伝送する場合の他のシステムの構成を表す図である。
【図２３】video_sequence関数のシンタックスを説明する図である。
【図２４】sequence_header()関数のシンタックスを説明する図である。
【図２５】sequence_extension()関数のシンタックスを説明する図である。
【図２６】extension_and_user_data(i)関数のシンタックスを説明する図である。
【図２７】user data()関数のシンタックスを表す図である。
【図２８】user dataに記述される関数のData_IDを説明する図である。
＊【図２９】V-Phase()関数のシンタックスを説明する図である。
【図３０】H-Phase()関数のシンタックスを説明する図である。
【図３１】Picture_Order()関数のシンタックスを説明する図である。
【図３２】Ancillary_data()関数のシンタックスを説明する図である。
【図３３】group_of_picture_header()関数のシンタックスを説明する図である。
【図３４】picture_header()関数のシンタックスを説明する図である。
【図３５】picture_coding_extension()関数のシンタックスを説明する図である。
【図３６】picture_data()関数のシンタックスを説明する図である。
【図３７】MPEG方式のレイヤを説明する図である。
【符号の説明】
４２，４３　MPEGエンコーダ，　４４，４５　MPEGデコーダ，　４６乃至４９SDTI-CPインタフェース，　５０　SDTI-CPネットワーク，　５１　SDTI-CPインタフェース，　６１，６２　TS MUX/DEMUX，　７２，７３　MPEGエンコーダ，７４，７５　MPEGデコーダ，　７６乃至８１　SDTI-CPインタフェース，　８０　SDTI-CPネットワーク

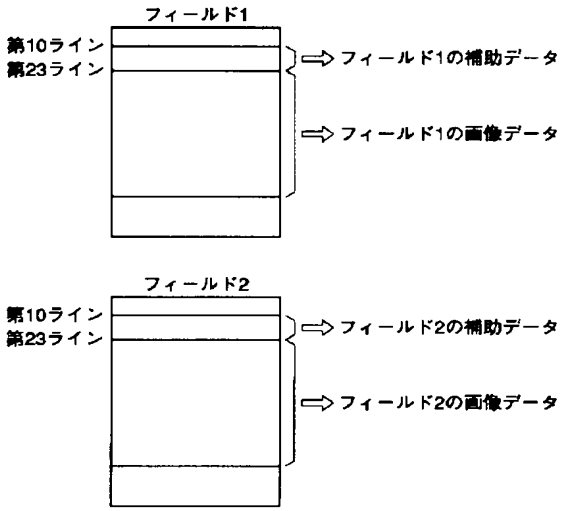
【図１】



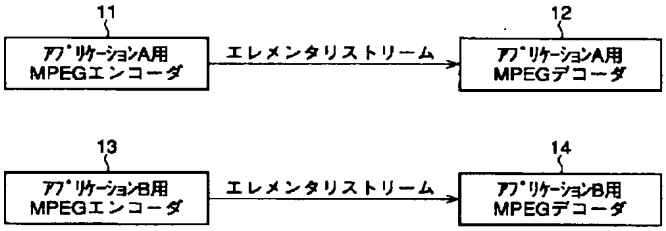
【図７】



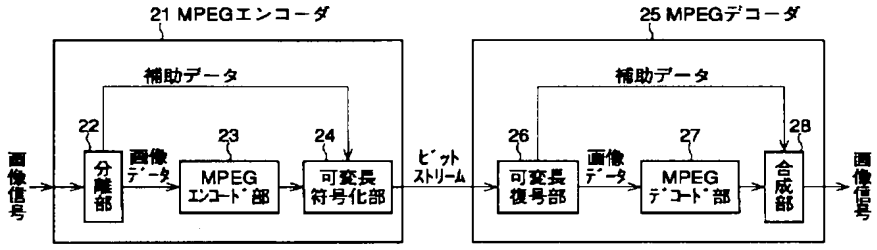
【図２】



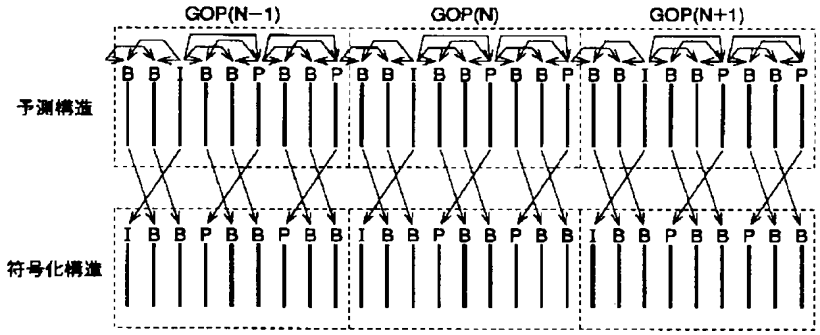
【図３】



【図４】



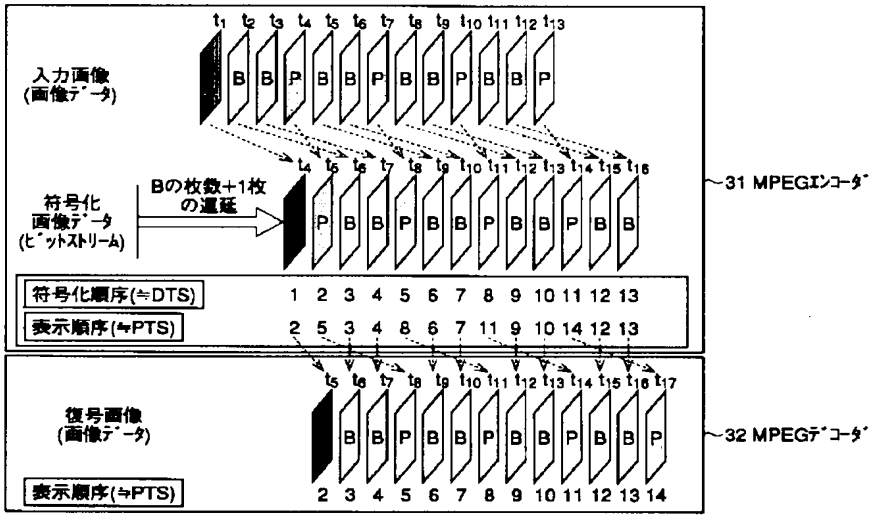
【図５】



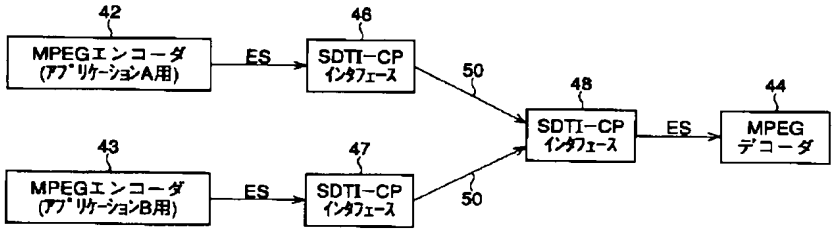
【図８】

SDTI−CP：放送局内

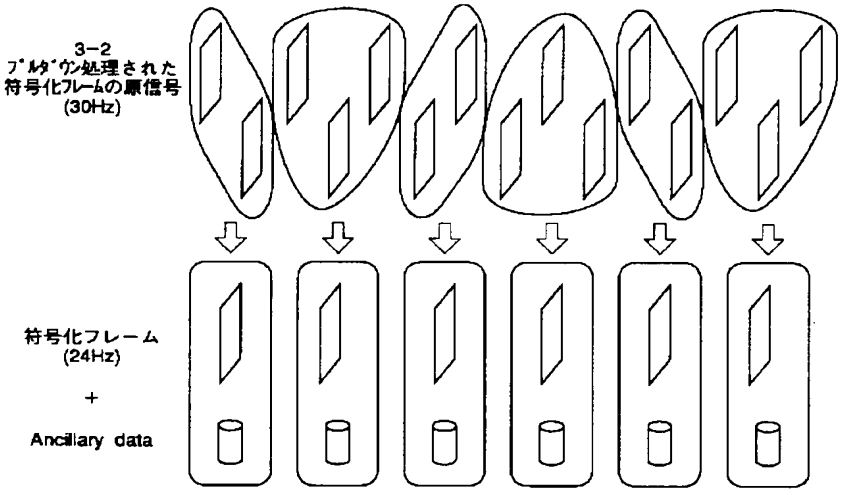
【図６】


入力画像
(画像ﾃﾞｰﾀ)
B B P B B P B B P B B P
符号化
画像ﾃﾞｰﾀ
(ﾋﾞｯﾄｽﾄﾘｰﾑ)
Bの枚数+1枚
の遅延
P B B P B B P B B P B B
31 MPEGｴﾝｺｰﾀﾞ
符号化順序(≒DTS)
1 2 3 4 5 6 7 8 9 10 11 12 13
表示順序(≒PTS)
2 5 3 4 8 6 7 11 9 10 14 12 13
復号画像
(画像ﾃﾞｰﾀ)
B B P B B P B B P B B P
32 MPEGﾃﾞｺｰﾀﾞ
表示順序(≒PTS)
2 3 4 5 6 7 8 9 10 11 12 13 14

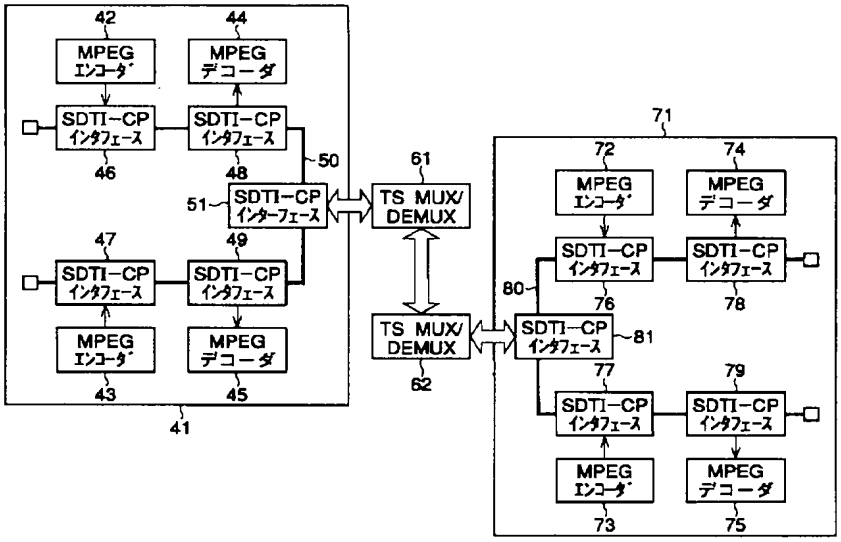
【図９】


42
MPEGエンコーダ
(ｱﾌﾟﾘｹｰｼｮﾝA用)
ES
46
SDTI-CP
ｲﾝﾀﾌｪｰｽ
50
48
SDTI-CP
ｲﾝﾀﾌｪｰｽ
ES
44
MPEG
ﾃﾞｺｰﾀﾞ
43
MPEGエンコーダ
(ｱﾌﾟﾘｹｰｼｮﾝB用)
ES
47
SDTI-CP
ｲﾝﾀﾌｪｰｽ
50

【図１０】


3-2
ﾌﾟﾙﾀﾞｳﾝ処理された
符号化ﾌﾚｰﾑの原信号
(30Hz)
符号化フレーム
(24Hz)
+
Ancillary data

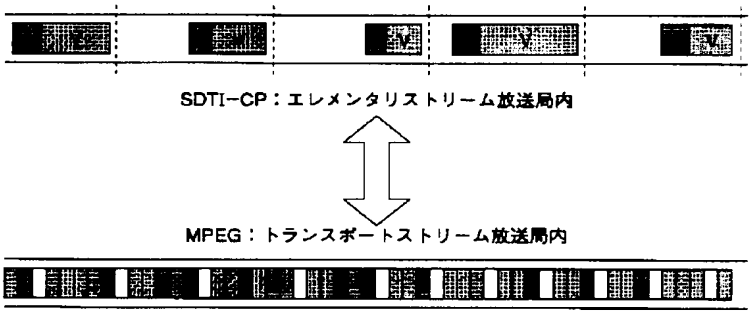
【図１１】



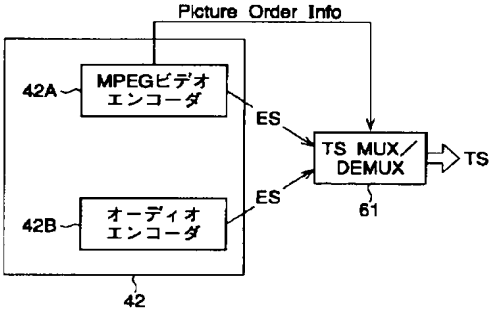
【図２８】

Data_ID	Data_type
00	FORBIDDEN
01	V−Phase
02	H−Phase
03	Time code 1
04	Time code 2
05	Picture Order
06	Video Index
07	Ancillary data
08	History data
...	...
80	Control flags
...	...
FF	User data

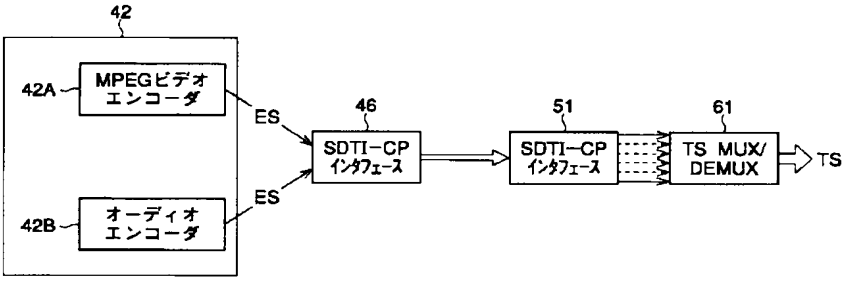
【図１２】



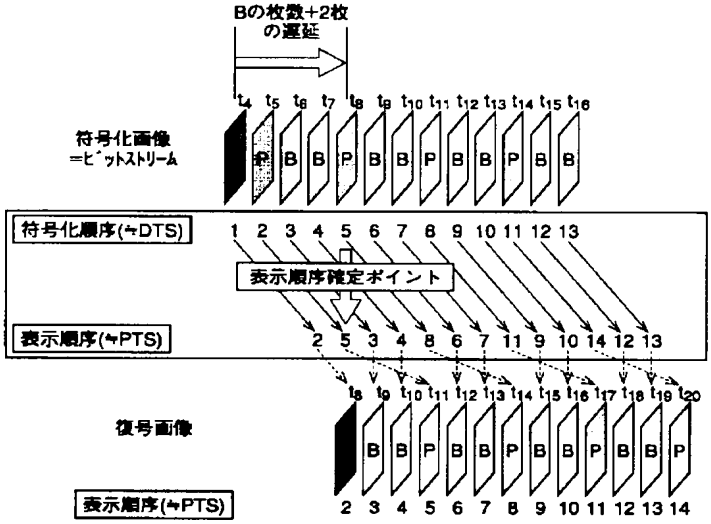
【図２２】



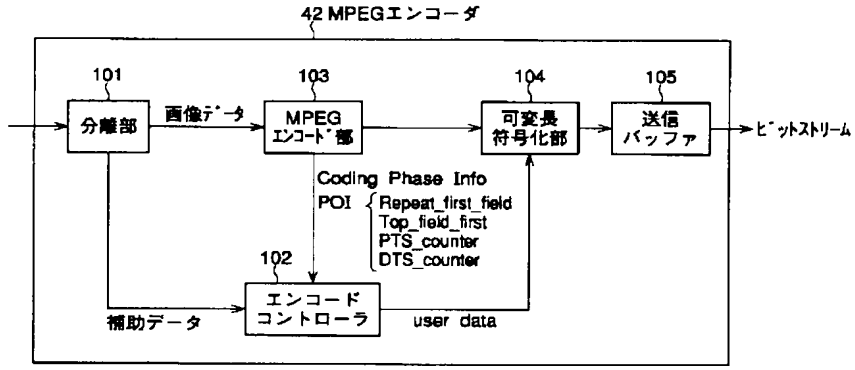
【図１３】



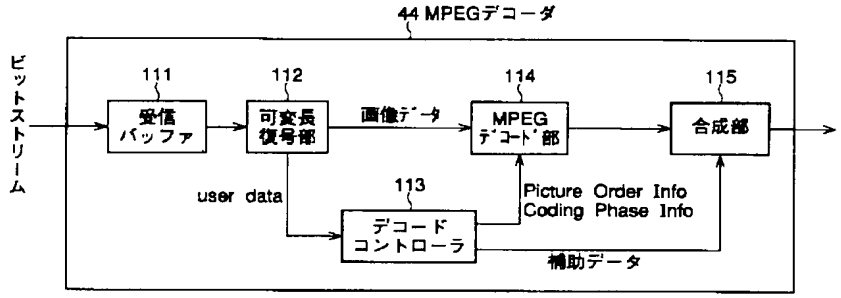
【図１４】


Bの枚数+2枚
の遅延
符号化画像
=ビットストリーム
符号化順序(≒DTS)
表示順序確定ポイント
表示順序(≒PTS)
復号画像
表示順序(≒PTS)

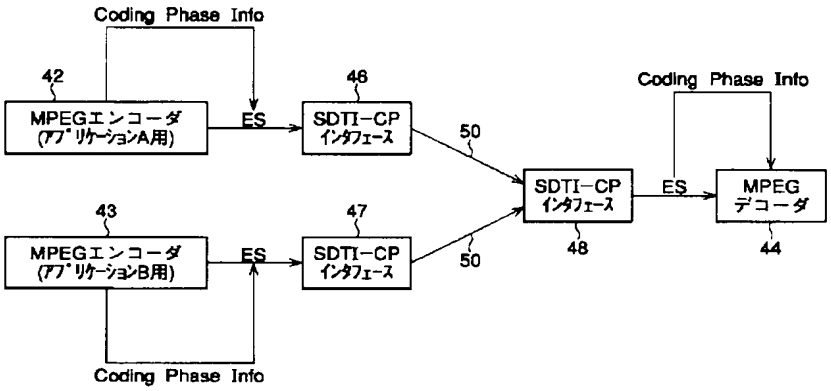
【図１５】


42 MPEGエンコーダ
分離部
画像データ
MPEG
エンコード部
可変長
符号化部
送信
バッファ
ビットストリーム
Coding Phase Info
POI
Repeat_first_field
Top_field_first
PTS_counter
DTS_counter
エンコード
コントローラ
補助データ
user data

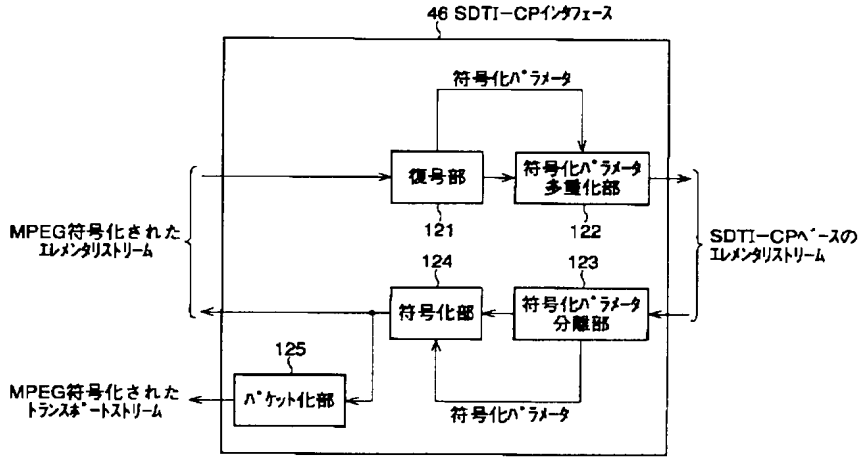
【図１６】


44 MPEGデコーダ
ビットストリーム
受信
バッファ
可変長
復号部
画像データ
MPEG
デコード部
合成部
user data
デコード
コントローラ
Picture Order Info
Coding Phase Info
補助データ

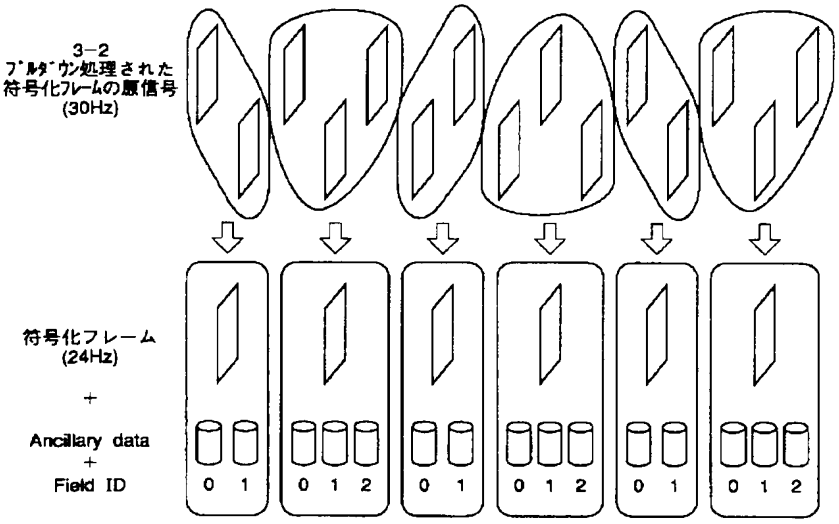
【図１７】


Coding Phase Info
42
MPEGエンコーダ
(アプリケーションA用)
ES
46
SDTI-CP
インタフェース
50
Coding Phase Info
SDTI-CP
インタフェース
ES
MPEG
デコーダ
48
44
43
MPEGエンコーダ
(アプリケーションB用)
ES
47
SDTI-CP
インタフェース
50
Coding Phase Info

【図１８】


46 SDTI-CPインタフェース
符号化パラメータ
復号部
符号化パラメータ
多重化部
121
122
MPEG符号化された
エレメンタリストリーム
SDTI-CPベースの
エレメンタリストリーム
124
123
符号化部
符号化パラメータ
分離部
125
MPEG符号化された
トランスポートストリーム
パケット化部
符号化パラメータ

【図１９】


3-2
プルダウン処理された
符号化フレームの原信号
(30Hz)
符号化フレーム
(24Hz)
+
Ancillary data
+
Field ID
0 1
0 1 2
0 1
0 1 2
0 1
0 1 2

【図２０】

Frame No	1	2	3	4	5	6	7	8	9	10	11	12	13
Picture type	I	B	B	P	B	B	P	B	B	I	B	B	P
Repeat_first_field	1	0	1	0	1	0	1	0	1	0	1	0	1
Top_field_first	1	0	0	1	1	0	0	1	1	0	0	1	1

(A)

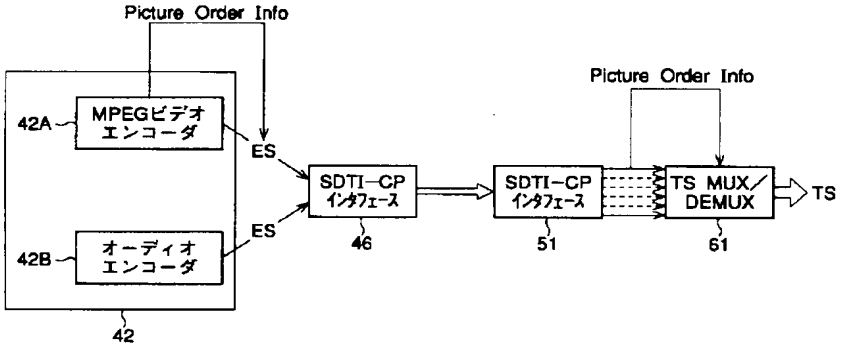
Frame No	1	2	3	4	5	6	7	8	9	10	11	12	13
Picture type	I	B	B	P	B	B	P	B	B	I	B	B	P
Repeat_first_field	1	0	1	0	1	0	1	0	1	0	1	0	1
Top_field_first	1	0	0	1	1	0	0	1	1	0	0	1	1
PTS_counter	0	3	5	8	10	13	15	18	20	23	25	28	30

(B)

Frame No	1	4	2	3	7	5	6	10	8	9	13	11	12
Picture type	I	P	B	B	P	B	B	I	B	B	P	B	B
Repeat_first_field	1	0	0	1	1	1	0	0	0	1	1	1	0
Top_field_first	1	1	0	0	0	1	0	0	1	1	1	0	1
DTS_counter	125	0	3	5	8	10	13	15	18	20	23	25	28

(C)

【図２１】



【図２３】

video_sequence(){	No. of bits	Mnemonic
next_start_code()		
sequence_header()		
sequence_extension()		
do{		
extension_and_user_data(0)		
do{		
if(nextbits()==group_start_code){		
group_of_pictures_header()		
extension_and_user_data(1)		
}		
picture_header()		
picture_coding_extension()		
extension_and_user_data(2)		
picture_data()		
}while((nextbits()==picture_start_code) |		
(nextbits()==group_start_code))		
if(nextbits()!=sequence_end_code){		
sequence_header()		
sequence_extension()		
}		
}while(nextbits()!=sequence_end_code)		
sequence_end_code	32	bslbf
}		

【図２６】

extension_and_user_data(i){	No. of bits	Mnemonic
while((i!=1)||(nextbits()&&(nextbits()==extension_start_code)) |		
(nextbits()==user_data_start_code)){		
if(nextbits()==extension_start_code)		
extension_data(i)		
if(nextbits()==user_data_start_code)		
user_data()		
}		
}		

【図２９】

Syntax	Bits	Mnemonic
V−Phase(){		
Data_ID	8	bslbf
V−Phase	16	uimsbf
}		

【図２４】

sequence_header(){	No. of bits	Mnemonic
sequence_header_code	32	bslbf
horizontal_size_value	12	uimsbf
vertical_size_value	12	uimsbf
aspect_ratio_information	4	uimsbf
frame_rate_code	4	uimsbf
bit_rate_value	18	uimsbf
marker_bit	1	"1"
vbv_buffer_size_value	10	uimsbf
constrained_parameters_flag	1	
load_intra_quantiser_matrix	1	
If(load_intra_quantiser_matrix)		
Intra_quantiser_matrix [64]	8*64	uimsbf
load_non_intra_quantiser_matrix	1	
If(load_non_intra_quantiser_matrix)		
non_intra_quantiser_matrix [64]	8*64	uimsbf
next_start_code()		
}		

【図２５】

sequence_extention(){	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
profile_and_level_indication	8	uimsbf
progressive_sequence	1	uimsbf
chroma_format	2	uimsbf
horizontal_size_extension	2	uimsbf
vertical_size_extension	2	uimsbf
bit_rate_extension	12	uimsbf
marker_bit	1	bslbf
vbv_buffer_size_extension	8	uimsbf
low_delay	1	uimsbf
frame_rate_extension_n	2	unimsbf
frame_rate_extension_d	5	uimsbf
next_start_code()		
}		

【図２７】

user_data(){	No. of bits	Mnemonic
user_data_start_code	32	
if(i--0){		
while(nextbits()!= "0000 0000 0000 0000 0000 0001"){		
If(nextbits()== "V-phase")		
V-phase()		
Else If(nextbits()== "H-phase")		
H-phase()		
{		
{		
Else if(i--2){		
while(nextbits()!= "0000 0000 0000 0000 0000 0001"){		
If((nextbits()== "Time code 1") || (nextbits()== "Time code 2"))		
Time_code()		
Else If(nextbits()== "Picture Order")		
Picture_order()		
{		
while(nextbits()!= "0000 0000 0000 0000 0000 0001"){		
If(nextbits()== "Ancillary_data")		
Ancillary_data()		
Else If(nextbits()== "History data")		
History data()		
{		
{		
If(nextbits()== "User data")		
user_data		
}		

【図３０】

Syntax	Bits	Mnemonic
H-Phase(){		
Data_ID	8	bslbf
H-Phase	8	uimsbf
}		

【図３２】

Syntax	Bits	Mnemonic
Ancillary_data(){		
Data_ID	8	bslbf
Field_ID	2	bslbf
Line_number	14	uimsbf
Ancillary_data_length	16	uimsbf
Marker_bits	1	bslbf
For(j=0; j<Ancillary_data_length; j++){		
Ancillary_data_payload	22	uimsbf
Marker_bits	1	bslbf
}		
While(!bytealigned())		
Zero_bit	1	"0"
}		

【図３１】

Syntax	Bits	Mnemonic
Picture_order(){		
Data_ID	8	bslbf
DTS_presence	1	bslbf
PTS_counter	7	uimsbf
If(DTS_presence== "1"){		
Marker_bits	1	bslbf
DTS_counter	7	uimsbf
}		
}		

【図３３】

group_of_picture_header(){	No. of bits	Mnemonic
group_start_code	32	bslbf
time_code	25	bslbf
closed_gop	1	uimsbf
broken_link	1	uimsbf
next_start_code()		
}		

【図３４】

picture_header(){	No. of bits	Mnemonic
picture_start_code	32	bslbf
temporal_reference	10	uimsbf
picture_coding_type	3	uimsbf
vbv_delay	16	uimsbf
if(picture_coding_type==2 | picture_coding_type==3){		
full_pel_forward_vector	1	
forward_f_code	3	uimsbf
}		
if(picture_coding_type==3){		
full_pel_backward_vector	1	
backward_f_code	3	uimsbf
}		
while(nextbits()== "1"){		
extra_bit_picture／*with the value "1" */	1	uimsbf
extra_information_picture	8	
}		
extra_bit_picture／*with the value "0" */	1	uimsbf
next_start_code()		
}		

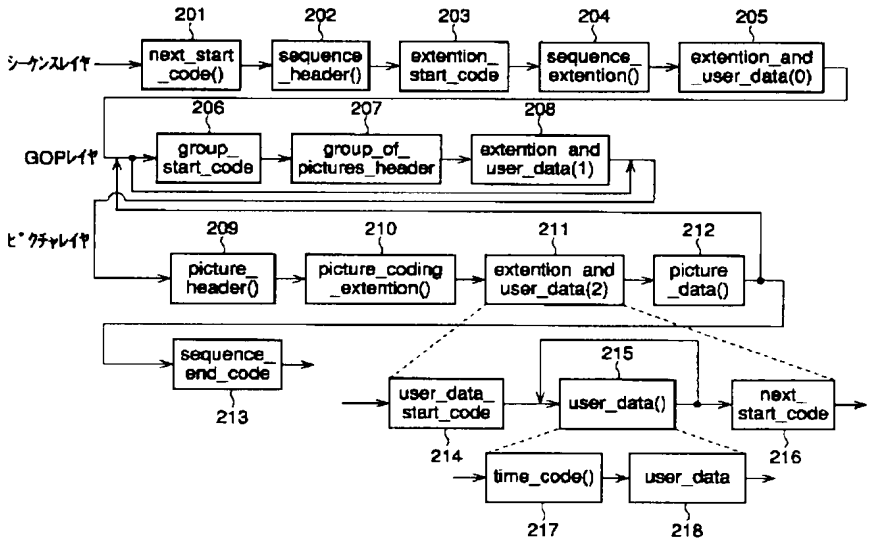
【図３５】

picture_coding_extension(){	No. of bits	Mnemonic
extension_start_code	32	bslbf
extension_start_code_identifier	4	uimsbf
f_code [0][0]／*forward horizontal*／	4	uimsbf
f_code [0][1]／*forward vertical*／	4	uimsbf
f_code [1][0]／*backward horizontal*／	4	uimsbf
f_code [1][1]／*backward vertical*／	4	uimsbf
intra_dc_precision	2	uimsbf
picture_structure	2	uimsbf
top_field_first	1	uimsbf
frame_pred_frame_dct	1	uimsbf
concealment_motion_vectors	1	uimsbf
q_scale_type	1	uimsbf
intra_vlc_format	1	uimsbf
alternate_scan	1	uimsbf
repeat_first_field	1	uimsbf
chroma_420_type	1	uimsbf
progressive_frame	1	uimsbf
composite_display_flag	1	uimsbf
if(composite_display_flag){		
v_axis	1	uimsbf
field_sequence	3	uimsbf
sub_carrier	1	uimsbf
burst_amplitude	7	uimsbf
sub_carrier_phase	8	uimsbf
}		
next_start_code()		
}		

【図３６】

picture_data(){	No. of bits	Mnemonic
do{		
slice()		
}while(nextbits()==slice_start_code)		
next_start_code()		
}		

【図３７】



フロントページの続き

Ｆターム(参考)　5C059 KK11 LA09 MA00 ME01 PP05
PP06 PP07 RB02 RB10 RB11
RC04 RC26 SS06 SS08 SS11
UA02 UA05 UA32